Departamento de Telemática

PROYECTO FIN DE CARRERA

Desarrollo de una aplicación web de Vídeo-llamada basada en SIP y Web-RTC

Autor: Alexandra Martínez Bowen
Tutor: Iván Vidal Fernández

Leganés, 28 de Enero de 2016

Título: Desarrollo de una aplicación de video-llamada basada en SIP y web-RTC
Autor: Alexandra Martínez Bowen
Director: Iván Vidal Fernández

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 28 de Enero de 2016 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Resumen

Hoy en día vivimos en un mundo en el que la comunicación se ha convertido en una necesidad de primer orden. Desde hace algunos años el concepto de comunicación se ha transformado radicalmente, vivimos conectados. Desde que suena el despertador por la mañana, hasta las últimas horas del día, nuestro *Smartphone* o tablet nos acompaña, proporcionándonos una conexión constante a Internet. Alrededor del 79% de los hogares españoles [29] y prácticamente el 100% de las empresas disponen de acceso a Internet. Esta transformación de la que hablamos no es debida sólo a la integración de Internet en nuestra vida diaria, también viene provocada por una continua evolución de las redes de telecomunicaciones, en especial las redes de datos, que han visto cómo su tráfico global aumentaba desde 100 GB por día en 1992 hasta 16.144 GB por segundo en 2014 [31], y las aplicaciones que utilizan dichas redes. En este contexto, cada vez adquieren mayor relevancia las aplicaciones multimedia, como origen del tráfico de datos en Internet. Hoy en día es muy común poder descargar un vídeo de *Youtube*, escuchar la radio a través de aplicaciones que utilizan la red de datos o utilizar aplicaciones como *Skype* para mantener videollamadas. Todo esto ha promovido el desarrollo de nuevas tecnologías enfocadas al mundo multimedia. Entre ellas se encuentra WebRTC (*Web Real Time Comunication*), una tecnología emergente que pretende estandarizar la comunicación multimedia entre navegadores.

El objetivo de este Proyecto Fin de Carrera consiste en el desarrollo de una aplicación web de videollamada basada en SIP (*Session Initiation Protocol*) y WebRTC. Además, esta aplicación deberá ser compatible con los procedimientos que han sido definidos por el 3GPP (*3rd Generation Partnership Project*) para el registro de usuarios y el establecimiento de sesiones multimedia, de modo que pueda operar en entornos de red con plano de control IMS (IP Multimedia Subsystem). La aplicación será accesible desde un navegador web *Google Chrome*. Finalmente, el entorno de pruebas utilizado ha contado con una infraestructura IMS real, la cual ha permitido verificar el correcto funcionamiento del desarrollo realizado.

Palabras clave: WebRTC (Web Real Time Comunication), IMS (IP Multimedia Subsystem), SIP (Session Initiation Protocol), Videollamada.

Abstract

Nowadays we live in a world where communication has become a necessity of the first order. In recent years the concept of communication has changed radically, we live connected. Since the alarm sounds in the morning until last hours in the day, our Smartphone or tablet are with us, providing a constant Internet connection. About 79% of Spanish households [29] and virtually 100% of companies have Internet access. This transformation of which we speak is not due only to the integration of Internet in our daily life, also it is caused by continuous development of telecommunications networks, especially data networks, which have seen their global traffic increased from 100 GB per day in 1992 to 16,144 GB per second in 2014 [31], and the applications that use these networks. In this context, increasing importance acquire multimedia applications such as source data traffic on the Internet. Today it is very common to download video from Youtube, listen to the radio through applications that use the data network or use applications like Skype to keep video calls. All this has promoted the development of new technologies focused on multimedia world. This includes WebRTC (Web Real Time Comunication), an emerging technology that aims to standardize multimedia communication between browsers.

The goal of this project is to develop of a video call web application based on SIP(Session Initiation Protocol) and WebRTC. In addition, this application must be consistent with the procedures that have been defined by 3GPP (3rd Generation Partnership Project) for the registration of users and the establishment of multimedia sessions, so that it can operate in a network environment with IMS Control plane (IP Multimedia Subsystem). The application will be accessible from a web browser Google Chrome. Finally, the test environment used has had a real IMS infrastructure, which has enabled verify proper operation of the development made.

Keywords: WebRTC (Web Real Time Comunication), IMS (IP Multimedia Subsystem), SIP (Session Initiation Protocol), Video call.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Motivación

Desde hace menos de una década los avances tecnológicos se han incrementado de forma exponencial, hasta llegar al día de hoy en el que la inclusión de las tecnologías en nuestra vida la ha transformado radicalmente, cambiando uno de los conceptos básicos para el ser humano, la comunicación. Hoy en día tenemos la necesidad de vivir conectados y tener acceso a todo tipo de datos en cualquier momento. El concepto de comunicación o conexión lo tenemos completamente integrado en nuestra vida diaria. Nuestro *Smartphone* siempre nos acompaña y tener acceso a Internet se ha vuelto una necesidad básica más. Esto queda reflejado en que alrededor del 79% de los hogares españoles tienen acesso a la red [29] y los usuarios de *Smartphone* experimentan un continuo crecimiento en España [30]. Esta necesidad de conexión la encontramos tanto en el mundo laboral como fuera de él. Cada día es más común mantener reuniones de trabajo con gente de cualquier punto del planeta por medio de cualquier tipo de aplicación de videollamada como *Skype, Hangouts, Facebook Messenger* o similares. Este tipo de aplicaciones aportan muchos beneficios a las empresas ya que anteriormente en empresas multinacionales era necesario el desplazamiento de gente de un sitio a otro para mantener reuniones presenciales, mientras que todo esto se simplifica con la conexión a través de la red. Otro beneficio destacado es el ahorro económico que supone la utilización de aplicaciones de videollamada.

Por otro lado, aparte de la inclusión de Internet en nuestra vida diaria, las redes de telecomunicaciones están en un proceso de cambio continuo. Ya no vale únicamente con

proporcionar un servicio de comunicaciones de cara al cliente, si no que este servicio prestado debe tener una calidad de experiencia mínima y ofrecer garantías de seguridad y ejecución de cara al usuario final. Las redes desarrolladas sobre tecnología IP soportan la provisión de estos requisitos, y a este respecto, la arquitectura de control IMS (*IP Multimedia Subsystem*) es a día de hoy una tecnología de uso extendido por parte de las grandes operadoras de telecomunicaciones, como por ejemplo Vodafone, Telefónica u Orange entre otros, para el soporte de los mismos. Así, IMS se ha convertido en la arquitectura de referencia para ofrecer servicios multimedia a través de redes IP de operador. Esta arquitectura cuenta con uno de los protocolos de señalización más sólidos y extendidos que encontramos actualmente, SIP (*Session Initiation Protocol*). Dicho protocolo es utilizado para el establecimiento, modificación y terminación de sesiones multimedia en redes de paquetes.

Bajo esta demanda de una continua conexión de personas en cualquier punto del planeta con una garantía mínima de calidad se encuadra el desarrollo de este Proyecto Fin de Carrera. El objetivo que se busca con este proyecto es el estudio de una tecnología emergente como es WebRTC. La principal ventaja de dicha tecnología es que permite la comunicación en tiempo real entre navegadores Web. Tras este estudio se pretende implementar una pequeña aplicación web de videollamada que demuestre las capacidades y posibilidades de dicha tecnología, así como su integración en la red IMS.

1.2 Objetivos

El objetivo principal de este Proyecto Fin de Carrera es el desarrollo e implementación de una aplicación de videollamada basada en el protocolo SIP y la tecnología WebRTC. Profundizando en este objetivo podemos destacar los siguientes puntos a tener en cuenta para el diseño e implementación del sistema.

- Se realizará un estudio de la tecnología emergente WebRTC y de su potencial para establecer comunicación en tiempo real entre navegadores Web.

- La aplicación desarrollada para desmostrar las capacidades de WebRTC permitirá establecer una videollamada directamente entre dos navegadores.

- La aplicación tendrá dos planos diferenciados: un plano de control, que soportará el intercambio de mensajes de señalización SIP para el establecimiento de vídeollamadas entre los navegadores web de dos usuarios, usando para ello una red IMS; y un plano de usuario que posibilitará intercambio de la información multimedia correspondiente a la vídeollamada (vídeo y audio) entre dichos navegadores.

- La aplicación desarrollada debe ser compatible con los procedimientos de registro y de establecimiento de sesión de la red IMS, según han sido definidos por el 3GPP.

- Todo el desarrollo se realizará con herramientas de tipo *OpenSource*.

- La aplicación será acessible mediante un navegador web Google Chrome en su versión 29 o superiores. Esto facilita la ejecución de la aplicación al usuario ya que no es necesaria la instalación de ningún tipo de software específico.

1.3 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de los capítulos que la forman:

- **Capítulo 1. Introducción y objetivos**: se describe el marco en el que se encuadra el proyecto. Así como su motivación y los objetivos que se quieren conseguir con su implementación.

- **Capítulo 2 Estado del arte:** se describen el conjunto de tecnologías empleadas para el desarrollo de este proyecto.

- **Capítulo 3 Identificación de los requisitos del sistema:** se identifican los distintos planos que forman el sistema que se desea implementar, así como los requisitos y funciones que debe cumplir cada uno.

- **Capítulo 4 implementación:** En este apartado se describe el desarrollo de los elementos del sistema basados en los requisitos nombrados en el apartado anterior y utilizando las tecnologías que se describirán en el capítulo 2.

- **Capítulo 5 Pruebas:** se describe el conjunto de pruebas ejecutadas y los resultados obtenidos para la validación de la implementación realizada.

- **Capítulo 6 Conclusiones y líneas futuras:** se describen las conclusiones obtenidas tras el desarrollo de la aplicación de videollamada, así como las posibles líneas de trabajo futuras que se pueden seguir a partir de los resultados obtenidos.

- **Capítulo 7 Anexos:** Para facilitar la lectura de este documento se incluye en este apartado una ampliación de la información de lo tratado en el presente documento. Se incluyen los siguientes anexos:

    - Anexo A: Presupuesto
    - Anexo B: Atributos opcionales en sesión SDP
    - Anexo C: Ofertas y Respuestas SDP
    - Anexo D: Glosario de términos

- **Capítulo 8 Bibliografía:** Se citan los principales libros y textos utilizados para la redacción del presente proyecto fin de carrera, así como las RFCs y documentos técnicos empleados.

Capítulo 2

Estado del arte

2.1 Introducción

La continua evolución de las comunicaciones en tiempo real junto con el incremento del uso de las redes de datos han provocado en el mercado una evolución de las redes móviles y de las tecnologías que utilizan. En este marco se encuadran los principales puntos de este capítulo.

Por una parte, WebRTC es una tecnología emergente que ha sido desarrollada por el IETF *(Internet Engineering Task Force*) junto con el W3C (*World Wide Web Consortium*) para permitir la comunicación en tiempo real entre navegadores. En la sección 2.2 se explica de forma más detallada su funcionamiento.

Por otra parte IMS es la arquitectura de referencia para ofrecer servicios multimedia sobre una red IP. En la sección 2.3 se explica detalladamente su evolución, su arquitectura y los principales protocolos que utiliza, entre ellos el más destacado es SIP que será utilizado para la gestión de la señalización dentro de la red IMS. Este protocolo se caracteriza por la creación, modificación y finalización de sesiones multimedia. En la sección 2.4 se explica con detalle el funcionamiento de dicho protocolo.

En este capítulo se pretende dar una breve descripción de cada una de las tecnologías, protocolos y lenguajes de programación en los que se encuadra el sistema desarrollado. Los tres puntos principales de este capítulo serán WebRTC, IMS y SIP. Aparte de estos tres puntos también se describe el formato SDP (Session Description Protocol), y como éste puede ser utilizado para la negociación de las características de una comunicación multimedia. Por otro lado, se incluye información sobre el protocolo RTP (Real-time Transport Protocol), que es un protocolo de nivel de transporte que soporta funciones relacionadas con la entrega de datos con características en tiempo real, y sobre los protocolos ICE (Interactive Connectivity Establishment) y STUN (Session Traversal Utilities for NAT) en los que se apoyará WebRTC para su correcto funcionamiento.

Para el desarrollo del sistema final se han utilizado diversos lenguajes de programación como Javascript con tecnologías AJAX (Asynchronous Javascript And Xml), para la implementación de WebRTC, JAVA para el desarrollo de la señalización SIP mediante sus librerías JAIN-SIP y HTML (Hipertext Transfer Protocol) para el desarrollo de la interfaz gráfica.

2.2 Web-RTC

WebRTC (*Web Real Time Comunication*) [1] es un API (*Application Programing Interface*) que está siendo desarrollada por el IETF[2] en colaboración con el W3C [28] para estandarizar la comunicación en tiempo real, de voz y vídeo, entre navegadores.

Figura 1: Símbolo de web-RTC

Tradicionalmente la comunicación en tiempo real entre navegadores requería la instalación de *plug-ins,* los cuales debían ser descargados e instalados por separado en el navegador. Esto supone varios inconvenientes ya que los *plug-ins* son específicos para cada navegador y sistema operativo, lo que no proporciona una forma de comunicación universal y estandarizada. Otro inconveniente que conlleva la utilización de *plug-ins* es que su instalación y actualización es responsabilidad del usuario. Estas limitaciones quedan resueltas con WebRTC.

Como toda tecnología WebRTC necesita unos requisitos mínimos para poder ser utilizada:

- Tener acceso al flujo de audio, vídeo y datos.
- Obtener información de la red como la dirección IP y el puerto para poder intercambiarla con otros usuarios.
- Intercambiar información sobre los medios y la capacidad del cliente, tales como la resolución y codecs.
- Tener un protocolo de señalización para iniciar y terminar la comunicación, asi como para informar de los posibles errores que se puedan producir.

Para todo esto Web-RTC ha implementado tres APIs:

- **MediaStream**: Utilizada para tener acceso a los flujos de datos, tales como el micrófono para el audio y la cámara para el vídeo.
- **RTCPeerConnection**: Utilizada para intercambiar información a nivel de señalización entre usuarios.
- **RTCDataChannel**: Utilizada para intercambiar datos entre usuarios.

2.2.1 APIs Web-RTC

2.2.1.1 Media Stream

El API de *Media Stream* describe los flujos de datos de audio y/o vídeo, los métodos para trabajar con ellos, las limitaciones asociadas con este tipo de datos, las respuestas de error y éxito al usar los datos de manera asíncrona y los eventos que se generan durante el proceso.

El API está basado en la manipulación de elementos *MediaStream*, los cuales representan un flujo de audio y vídeo. Un *MediaStream* está compuesto por objetos *MediaStreamTrack* que representan varias pistas de audio o video. Cada *MediaStreamTrack* puede tener uno o más canales. El canal representa la menor unidad de flujo.

Los objetos *MediaStream* tiene una entrada y una salida, como podemos ver en la siguiente figura.

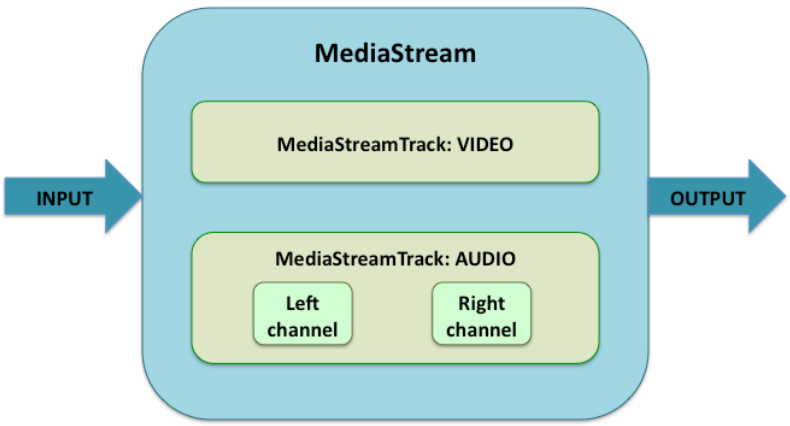


Figura 2: Objetos MediaStream

2.2.1.2 RTCPeerConnection

WebRTC utiliza el API de *RTCPeerConnection* para la transmisión eficiente de datos, audio y vídeo, entre los dos extremos de la comunicación, los dos navegadores implicados. Para un correcto funcionamiento de WebRTC se necesita un mecanismo para coordinar la comunicación y enviar mensajes de control, este mecanismo es conocido como señalización. La señalización es un conjunto de métodos y protocolos que no están especificados en el API de WebRTC. La elección del mecanismo de señalización de deja de la mano de los desarrolladores de aplicaciones basadas en WebRTC.

La señalización se utiliza fundamentalmente para el intercambio de tres tipos de información: Mensajes de control de la sesión, configuración de la red y capacidades de los extremos de la comunicación. Este intercambio de información debe completarse de forma satisfactoria antes de que la comunicación comience, y para que esto ocurra Web-RTC se apoya en dos protocolos de información de red ICE (*Interactive Connection Establishment)* y STUN (*Simple Transversal Utilities for NAT*), que se describirán brevemente en los apartados 2.7 y 2.8 del presente documento.

2.2.1.3 RTCDataChannel

Aparte de audio y video WebRTC, soporta otros tipos de comunicaciones en tiempo real. *RTCDataChannel* API [3] permite este intercambio bidireccional, extremo a extremo, con un bajo retardo y un alto rendimiento. Se puede utilizar en modo fiable con una velocidad mas lenta pero con entrega garantizada de paquetes o en modo no fiable para velocidades mas rápidas y asumiendo una posible pérdida de paquetes.

Este API tiene varias funciones para aprovechar al máximo los canales de comunicación establecidos y permitir una comunicación robusta y flexible extremo a extremo:

- Permite múltiples canales simultáneos con priorización.
- Seguridad integrada (DTLS *Datagram Transport Layer Security*) y control de congestión.
- Capacidad para ser utilizada con o sin audio y vídeo.
- Entrega fiable y no fiable.

Entre los múltiples usos que se le puede dar a esta funcionalidad destacan:

- Juegos en tiempo real
- Control remoto de escritorio
- Transferencia de archivos
- Aplicaciones de mensajería instantánea

2.2.2 Voice y Video Engine

En este apartado se analizarán los codecs que utiliza WebRTC para la correcta transmisión y recepción de audio y vídeo.

CODEC es la abreviatura de CODificador-DECodificador. Un codec es un conjunto de funciones algorítmicas que son necesarias para comprimir o codificar cualquier tipo de información en el origen y para descomprimirla o decodificarla en el destino. Cuando se intercambian archivos es necesario que tanto emisor como receptor tengan los mismos codecs ya que el proceso de codificación y decodificación debe seguir el mismo modelo para que el audio y vídeo transmitidos se presenten de forma íntegra en los dos extremos de la comunicación.

2.2.2.1 Codecs de audio

Los codecs de audio [6] son los responsables de que el audio se presente de forma íntegra de un extremo a otro extremo de la comunicación en las aplicaciones de WebRTC. Se encargan del tratamiento de audio y la cancelación de ecos. Algunos de los codecs de audio que utiliza WebRTC son:

- **iSAC** (*Internet Speech Audio Code*): es un codec de banda ancha desarrollado por el GIPS (*Global IP Solutions*). Fue adquirido por Google en 2011 para incorporarlo al proyecto de WebRTC [1]. Las características técnicas principales de este codec son: Frecuencia de muestreo de 32kHz, tasa de bit variables y adaptable de 10kbit/s a 52kbit/s, tamaño de la muestra adaptable entre 30ms y 60ms.

- **iLBC** (*Internet Low Bitrate Code*): es un codec de banda estrecha desarrollado por el GIPS. Fue adquirido por Google en 2011 para su incorporación al proyecto de WebRTC [1]. El principal atractivo de iLBC recae en su capacidad para mantener la calidad aunque se produzcan pérdidas de muestras. Las características técnicas principales de este codec son: Frecuencia de muestreo de 8kHz, respuesta controlada a la pérdida de paquetes, retraso y jitter, tasa de bits fija (15.2 kbit/s para muestras de 20 ms y 13.33 kbit/s para 30 ms), tamaño de muestra fijo (304 bits por bloque para muestras de 20 ms y 400 bits por bloque para muestras de 30 ms).

- **OPUS** [26]: es un compresor de audio con pérdidas de libre utilización desarrollado por la IETF. Este codec tiene un alto grado de ajuste, pudiendo elegir tasas de bits altas o bajas dependiendo de la calidad de servicio que queramos ofrecer. La frecuencia de muestreo se extiende entre 8kHz y 48kHZ abarcando todo el espectro audible humano y el tamaño de las muestras puede variar entre 2,5 y 60ms. Todas estas características pueden ser cambiadas en cualquier momento de la comunicación sin experimentarse una parada en el servicio.

2.2.2.2 Codecs de vídeo

Los Codecs de vídeo [7] permiten comprimir y descomprimir vídeo digital. Normalmente los algoritmos de compresión empleados conllevan una pérdida de información. El codec de vídeo seleccionado para el proyecto de WebRTC es VP8.

VP8 es creado por On2 Technologies, una empresa especializada en desarrollar tecnologías de compresión de vídeo. En 2009 esta empresa es comprada por Google y en 2010 el codec es incorporado al proyecto de WebRTC.

Desde el comienzo del desarrollo de VP8, los desarrolladores se centraron en aplicaciones de vídeo basados en web. El diseño de VP8 se hizo en base a las siguientes premisas:

- Bajo requerimiento de ancho de banda
- Hardware del cliente heterogéneo
- Formato de vídeo Web

Las principales características técnicas de este codec son:

- Transformada híbrida con cuantificación adaptativa.
- Referencias flexibles a fotogramas.
- Predicción Intra e Inter eficiente.
- Alto rendimiento en interpolación de Sub-Pixeles.
- Filtro de bucle adaptativo.
- Codificación de entropía adaptativo a nivel de marco.
- Partición de dato con procesamiento en paralelo.

2.2.3 Establecimiento de sesión en web-RTC

Como se ha comentado en el apartado 2.2 el objetivo principal de web-RTC es la estandarización de la comunicación en tiempo real entre navegadores. Si se analiza la evolución de las comunicaciones en la web desde sus inicios, Web-RTC supone un gran avance. En el inicio se tenía una comunicación *half-duplex* entre el navegador y un servidor por medio de HTTP (*HyperText Transfer Protocol*), mas tarde se consiguió una comunicación *full-duplex* entre el navegador y el servidor mediante *websocket* y WebRTC consigue una comunicación *full-duplex* directamente entre navegadores. Esto lo vemos ilustrado en la siguiente figura.

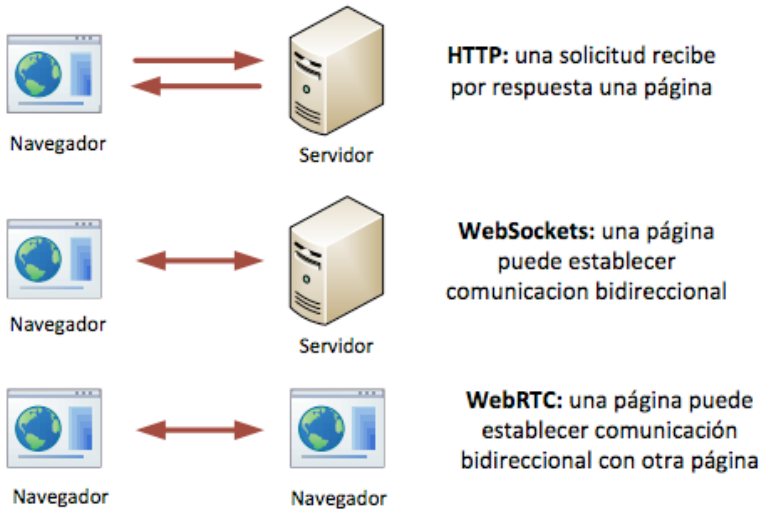


Figura 3: Evolución de las comunicaciones en la web

El modelo del navegador y su papel en la comunicación en tiempo real se muestra en la siguiente figura.

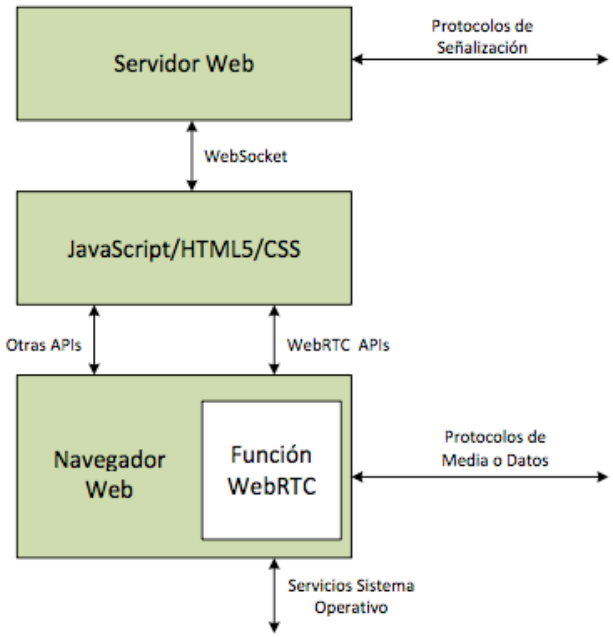


Figura 4: Modelo en el navegador [35]

Para el establecimiento de sesión entre navegadores Google ha desarrollado un modelo denominado JSEP (*Javascript Session Establishment Protocol*) en el que se han separado el plano de señalización y el plano de datos. Esta separación de los dos planos queda reflejado en la figura anterior. Por un lado el navegador tendrá implementada la mínima funcionalidad posible y sólo se encargará de la gestión de los datos, la parte de gestión de la señalización queda del lado del servidor en el nivel de aplicación. El nivel de aplicación se encarga de entregarle al navegador los parámetros necesarios para el establecimiento de la conexión. El nivel de aplicación es también el responsable de intercambiar la información necesaria con el otro lado de la comunicación, es decir, de la

gestión de la señalización. Como se ha dicho anteriormente WebRTC no define la señalización, deja esta parte a cargo de los desarrolladores, pero las opciones más comunes y extendidas para esta gestión de la señalización son mediante servidores SIP o *App Engine*.

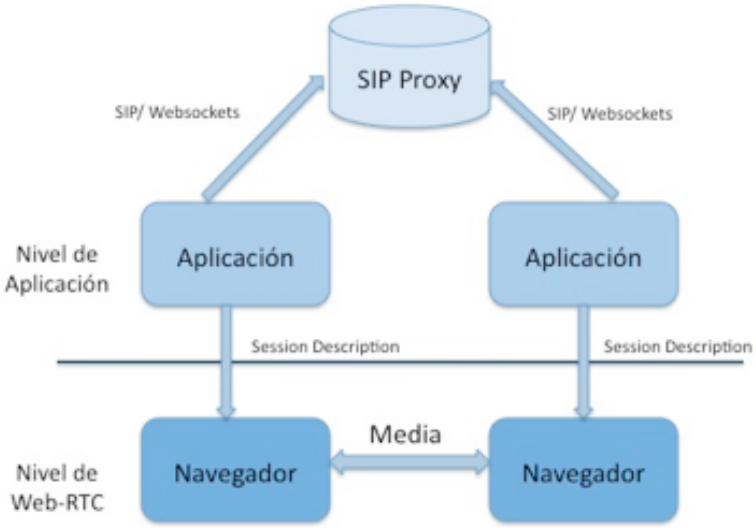


Figura 5: Implementación de WebRTC mediante SIP Proxy

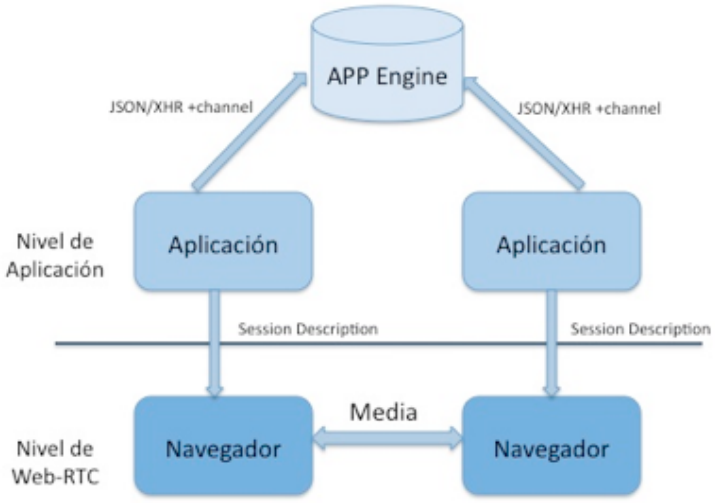


Figura 6: Implementación de WebRTC mediante APP Engine

Para el inicio de comunicación entre dos navegadores ha de seguirse el siguiente orden:

1. El navegador que inicia la comunicación debe tener acceso a su descripción local, esto es, tener acceso al audio y vídeo de su dispositivo y conocer sus codecs.

2. Esta descripción local del primer navegador se envía mediante el método de señalización elegido al segundo navegador.
3. La descripción local del primer navegador será establecida como descripción remota en el segundo navegador.
4. El segundo navegador debe tener acceso al audio y vídeo de su dispositivo y conocer los codecs que tiene implementados para formar su descripción local y enviarla mediante la señalización al primer navegador.
5. El primer navegador al recibir la descripción local del navegador dos la establece como descripción remota.
6. En este momento ya se ha intercambiado la información necesaria para el estabecimiento de la comunicación y ésta queda establecida.

Tras la definición de los pasos que han de seguirse para el establecimiento de sesión entre dos navegadores mediante WebRTC, este intercambio de mensajes queda representado a través de dos figuras geométricas, el triángulo donde ambos navegadores cuelgan del mismo servidor y el trapecio donde cada navegador cuelga de un servidor diferente. Esto queda representado en las siguientes figuras.

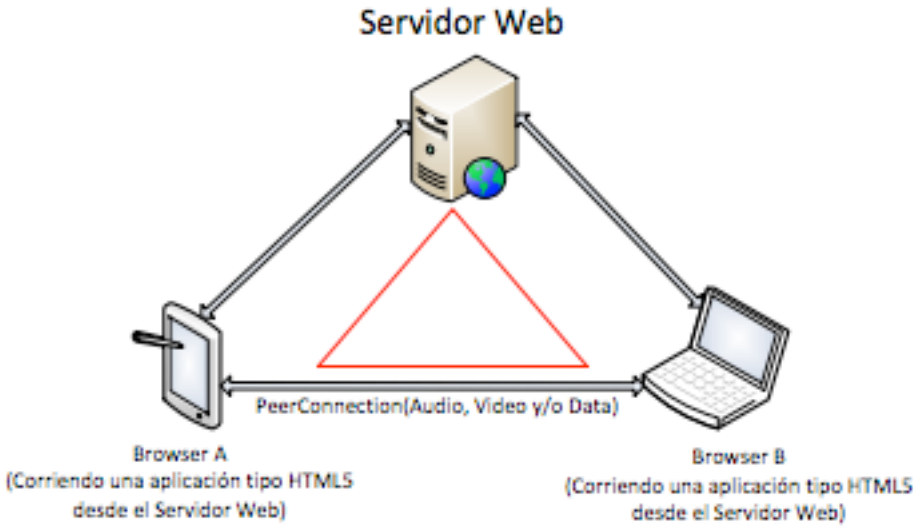


Figura 7: Comunicación WebRTC en triángulo [10]

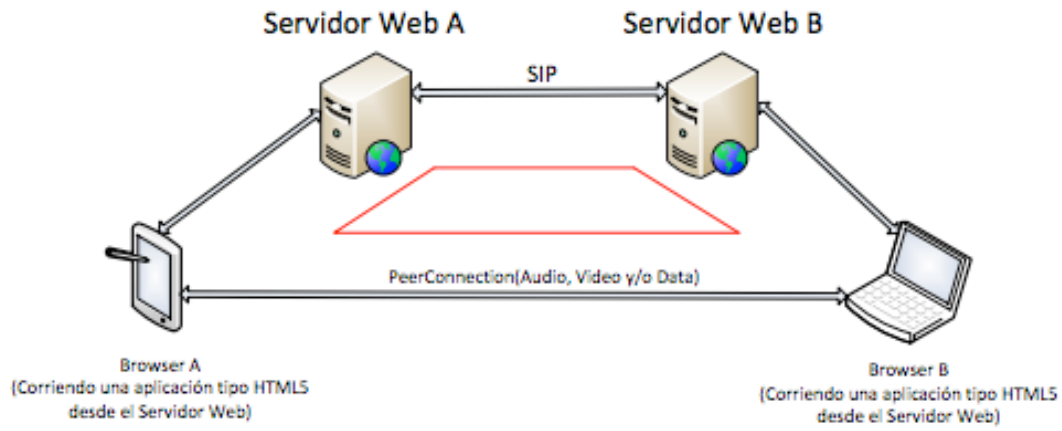


Figura 8: Comunicación WebRTC en trapecio [10]

2.2.4 Seguridad en Web-RTC

La seguridad en las comunicaciones en tiempo real o los *plug-ins* puede verse comprometida de varias formas. Por ejemplo:

- Los datos sin cifrar podrían ser interceptados en el camino entre navegadores o entre navegador y servidor.
- Una aplicación puede registrar y distribuir imágenes de vídeo o de audio sin que el usuario lo sepa.
- Junto a la actualización de los plugins puede instalarse en el ordenador algún virus o malware aunque aparentemente no se vea nada

WebRTC tiene varias características para evitar estos problemas:

- Implementaciones WebRTC utilizan protocolos seguros, como DTLS y SRTP.
- WebRTC no es un *plug-in*, sus componentes se ejecutan en el navegador y no en un proceso separado, los componentes no requieren de una instalación separada y se actualizan cada vez que se actualiza el navegador.
- Debe concederse explícitamente acceso a la cámara y micrófono del equipo y mostrarse claramente en la interfaz de usuario.

2.2.5 Web-RTC y VoIP

Hoy en día existe bastante confusión entre las similitudes y diferencias que se pueden encontrar entre el novedoso WebRTC y VoIP. En la siguiente tabla se encuentra una breve comparativa entre los sistemas tradicionales IP de comunicación en tiempo real y WebRTC.

Características	**Voip**	**WebRTC**
Señalización	SIP y H.323	Sin definir
Media	RTP/RTCP	RTP/RTCP
Codecs de audio	Serie G.7xx	OPUS, iSAC e iLBC
Codecs de vídeo	H.263 y H.264	VP8
Seguridad	SRTP/TLS/IPsec	SRTP y DTLS

Tabla 1: Comparativa entre VoIP y WebRTC

A diferencia de VoIP, que tiene como protocolos principales de señalización SIP y H.323, WebRTC no define la señalización, esto permite a los fabricantes poder utilizar WebRTC independientemente del protocolo de señalización empleado. Las opciones de señalización más populares, que no las únicas, son JSON (*JavaScript Over Notation*) sobre WebSockets y SIP sobre WebSockets.

Para la comunicación en tiempo real son imprescindibles los codecs de audio y vídeo y la capa de transporte. En cuanto a la capa de transporte tanto VoIP como WebRTC utilizan el conjunto de protocoles RTP/RTCP. Para los codecs de audio VoIP utiliza los tradiciones G.7xx y WebRTC OPUS principalmente, aunque también podemos encontrar los Codecs iSAC e iLBC. Tanto los Codecs G.7xx y OPUS no necesitan licencia, pero OPUS además es de código abierto, una razón muy atractiva para ser utilizado por los desarrolladores. En cuanto a los codecs de vídeo VoIP utiliza el codec más popular de la industria H.264 aunque tiene unos costes de licencia elevados. Por otro lado WebRTC apuesta por el nuevo codec VP8 enfocado principalmente hacia un formato de vídeo web. Aunque no está tan extendido e implementado como H.264 cada vez está mas extendido en la industria.

A parte de lo descrito anteriormente, la mayor diferencia es que WebRTC se centra exclusivamente en las comunicaciones entre navegadores, esto le permite por ejemplo utilizar de forma principal el CoDEC Vp8 para vídeo, mientras que VoIP tiene multitud de servicios como pueden ser voz sobre LTE (VoLTE) o *Rich Suite Comunication* (RCS) entre otros.

2.3 IMS

2.3.1 ¿Qué es IMS?

IMS [8] es una arquitectura de referencia para ofrecer servicios multimedia sobre una infraestructura IP. Estos servicios multimedia se extienden desde el vídeo en streaming, voz sobre IP (VoIP), el reciente voz sobre LTE (VoLTE), la videoconferencia o los servios *Push To Talk Over Cellular* (PoC). Se trata de una arquitectura de control especificada por primera vez en la *release* 5 de especificaciones del 3GPP [9] en colaboración con el IETF.

IMS se puede definir como la arquitectura de convergencia entre redes fijas y móviles. En la siguiente figura puede apreciarse lo que significa la convergencia en la red IMS. Antes de IMS todos los servicios estaban separados y existían servidores dedicados para cada función. Esto provocaba tener funciones replicadas ya que era necesario abrir una sesión por cada servicio que se utilizaba. Con la aparición de IMS los distintos servicios comienzan a integrarse, manteniendo una sesión única para todos y empleando servidores comunes.

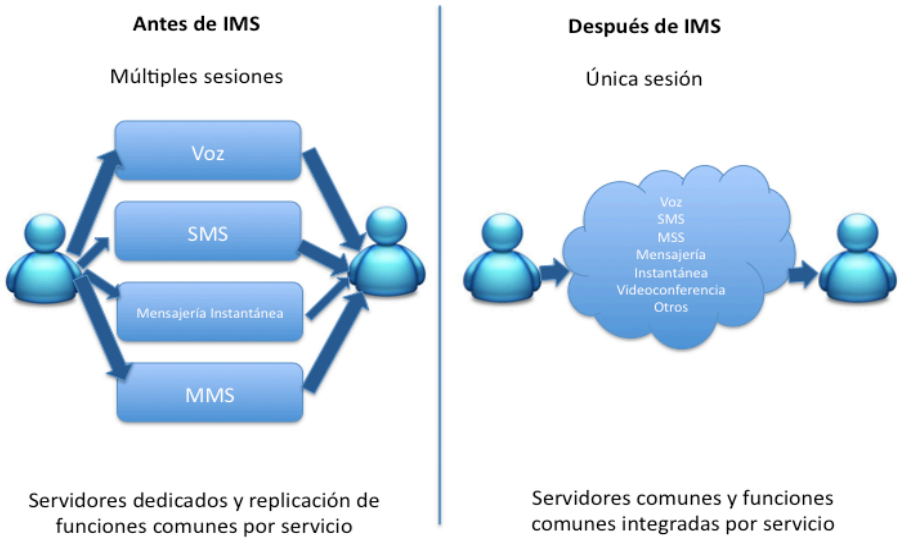


Figura 9: IMS: Arquitectura de convergencia

Como toda arquitectura, IMS tiene un conjunto requisitos básicos [36] para que el servicio pueda ser prestado como está especificado por el 3GPP:

- Conectividad IP: El usuario debe disponer de conectividad IP para poder utilizar los servicios que ofrece la red IMS.

- Independencia hacia el acceso: IMS está diseñado con independencia del acceso con el fin de que cualquier tipo de acceso(UMTS, DSL….) que utilice conectividad IP pueda disfrutar de los servicios ofrecidos por IMS.

- Garantía de calidad de los servicios multimedia: Las redes de acceso y de transporte de IMS ofrecen calidad de servicio desde el principio hasta el final de la cadena. A través de IMS, el terminal negocia sus capacidades y expresa sus exigencias de calidad de servicio durante la fase de establecimiento de sesión con el protocolo SIP.

- Control de políticas El control de políticas IP significa la capacidad de autorizar y controlar el uso del trafico a nivel de multimedia en IMS, sobre los parámetros de la señalización SIP intercambiados durante el establecimiento de la sesión.

- Comunicaciones segurizadas: IMS ofrece mecanismos de seguridad similares a los de GSM/GPRS (*Global System for Mobile Comunication/ General Packet Radio Services*), por ejemplo, asegurando que el usuario ha sido autentificado antes de acceder a los servicios.

- Tasación: IMS permite los modelos de pre-pago (*on-line*) y post-pago (*off-line*).

- Soporte de *Roaming:* El usuario podrá acceder a sus servicios IMS desde cualquier red IMS que no sea la propia.

- Interconexión con otras redes

- Control de servicios: los operadores pueden utilizar dos políticas de control de servicios, políticas generales aplicadas a todos los usuarios o políticas individuales aplicadas a cada usuario.

2.3.2 Origen y evolución de IMS

La evolución de la arquitectura de IMS ha sido desarrollada a través de las distintas releases del 3GPP y desde sus inicios la arquitectura de la red IMS no ha variado prácticamente.

IMS aparece por primera vez en la *release* 5 del 3GPP. En dicha release sólo se contempla a los usuarios de telefonía móvil, es decir, los que utilizan una red de acceso radio (RAN). En este tipo de acceso se encuentran los usuarios de UMTS, CDMA200 y GSM/GPRS, aunque estos últimos no aprovechan enteramente la capacidad de la red IMS debido al ancho de banda limitado con el que trabajan.

En la *release* 6 se incorpora el acceso para redes inalámbricas tales como WLAN y WIMAX. Es necesaria también la implementación de un gateway para este tipo de accesa, el cual es llamado *Wireless LAN Gateway (WAG),* su función es hacer que los datos de la red de acceso sean entendidos por el Core IMS.

La *release* 7 nace de la colaboración entre el 3GPP, 3GPP2 y la release 1 de TISPAN. En esta *release* se implementa todo el acceso de las redes fijas, principalmente las redes DSL y Ethernet, para brindar mejores servicios de ancho de banda, servicios de voz y servicios multimedia. Es necesario implementar un Servidor de Acceso de Banda Ancha (BAS) para comunicar los datos entre la red de acceso y el Core IMS.

2.3.3 Arquitectura de IMS

La arquitectura IMS puede ser estructurada en cuatro capas principalmente:

- **Capa de acceso**: Puede representar todo acceso de alta velocidad, tecnologías de acceso de banda ancha usadas por las redes móviles como UMTS, UTRAN, xDSL, WIFI…etc.

- **Capa de transporte**: Representa una red IP. Esta red IP integra mecanismos de calidad de servicio.

- **Capa de control**: Consiste en controladores de sesión responsables del encaminamiento de la señalización entre usuarios y de la invocación de los servicios.

- **Capa de aplicación o servicios**: Introduce las aplicaciones (servicios de valor añadido) propuestas a los usuarios.

En la siguiente figura ser pueden ver dichas capas

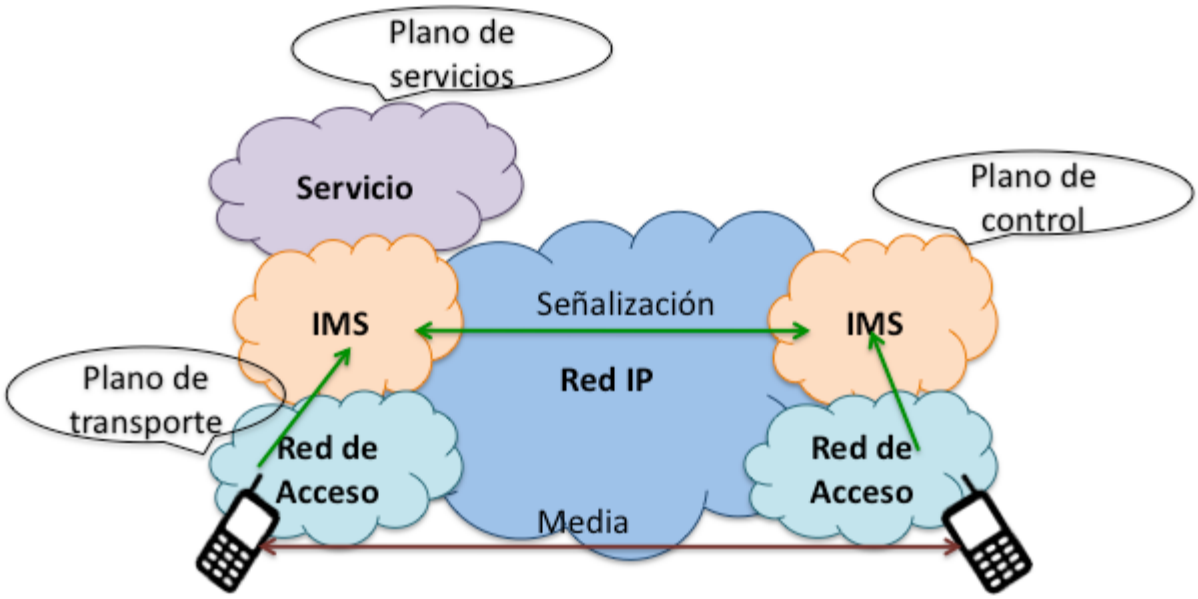


Figura 10: Capas arquitectura IMS

Dentro de la arquitectura cabe destacar los siguiente elementos según su funcionalidad:

- Elementos de almacenamiento de datos: HSS (Home Suscriber Server) y SLF (Suscription Location Function).
- Elementos de control: P-CSCF (Proxy Call State Control Function), I-CSCF (Interrogating Call State Control Function) y S-CSCF (Service Call State Control Function).
- Servidores de aplicación y pasarelas: SIP AS (SIP Aplication Server), OSA SCS (Open Service Architectur Service Capability Server) e IM-SSF(IP Multimedia Service Switching Function).
- Elementos de interconexión: MGW/MGCF (Media Gateway/ Media Gateway Control Function), SGW (Signalling Gateway) y BGCF (Breakout Gateway Control Function).
- Elementos multimedia: MRFC (Media Resource Function Controller) y MRFP (Media Resource Function Processor).

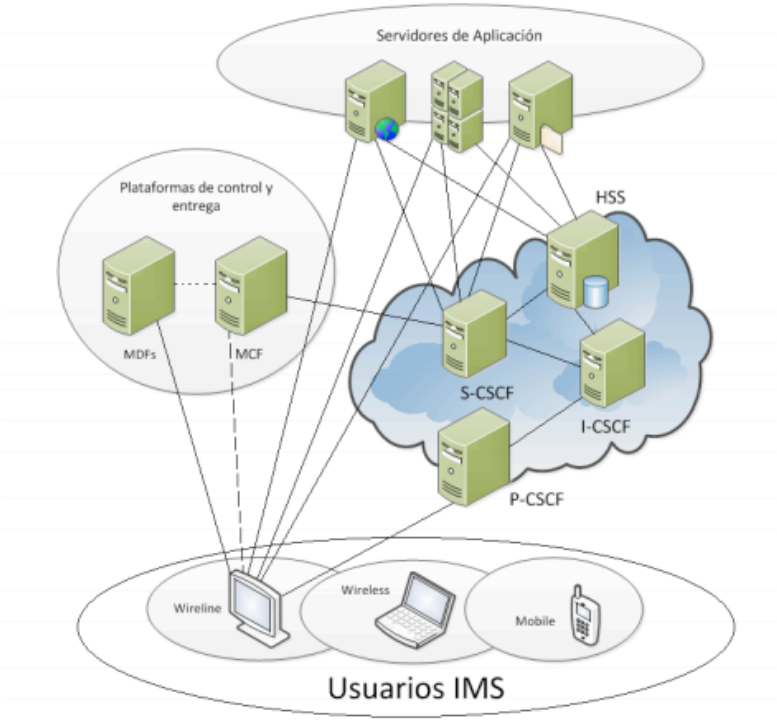


Figura 11: Arquitectura IMS [10]

2.3.3.1 HSS y SLF

El HSS (*Home Suscriber Server*) es la evolución del HLR (*Home Local Register*) de GSM. Es la base principal de almacenamiento de los datos de los usuarios y de los servicios a los cuales están suscritos. Contiene datos relacionados con el proceso de registro del usuarios, tales cómo el S-CSCF que tiene asignado el usuario y su dirección, la identidad de los usuarios y políticas de perfil del usuario entre otros.

Existen dos tipos de identidades de usuario, las llamadas públicas y las privadas. Las identidades públicas son las que los usuarios utilizan para establecer conexiones y las identidades privadas se utilizan para los procesos de registro y autenticación de usuarios. El HSS guarda la relación entre ambas en su base de datos.

El HSS utiliza el protocolo DIAMETER para comunicarse con el I-CSCF y el S-CSCF y los protocolos SIP/OSA para comunicarse con los servidores de aplicación. Para la comunicación con dichos servidores el HSS almacena información en los IFC (*Initial Filter Criteria*), que son los encargados de hacer un filtrado de la señalización SIP y encaminarlos al servicio de aplicación correspondiente.

El SLF (*Suscription Location Function*) es usado cuando la red tienes mas de un HSS para localizar el HSS que tiene asignado el usuario.

2.3.3.2 CSCFs

Los CSCFs (*Call State Control Function*) actúan como nodos de control en el plano de señalización. Existen tres tipos de CSCFs:

- P-CSCF (*Proxy-CSCF*) es el primer punto de contacto del terminal en el dominio de la red IMS. Su dirección es descubierta por el terminal de usuario durante la activación de un contexto PDP para el intercambio de mensajes de señalización SIP. El P-CSCF actúa como un un servidor proxy de SIP encaminando los mensajes SIP hacia el destinatario apropiado.

- S-CSCF (*Serving-CSCF*) es el encargado del control de la sesión. Durante el proceso de registro a cada usuario se le asigna un S-CSCF que se encargará, como ya hemos dicho, del control de la sesión, la autenticación del usuario, aplica las políticas del operador de red y aplica los perfiles del usuario. Interactúa con el HSS/SLF para obtener la información del usuario y con el AS para la provisión de servicios. El S-CSCF debe encontrarse en la red origen del usuario y nunca en una red visitada.

- I-CSCF(*Interrogating-CSCF*) es el punto de entrada a la red IMS para redes externas. La dirección IP del I-CSCF es obtenida mediante una consulta DNS. El I-CSCF actúa como servidor Proxy de SIP y oculta la topología de la red hacia el exterior. Interactúa con el HSS y SLF para tareas de autenticación de usuarios. Selecciona y asigna el S-CSCF al usuario con la información obtenida del HSS durante el registro.

2.3.3.3 Servidores de aplicación y pasarelas

Los servidores de aplicación, AS (*Aplication Server*), son los nodos donde se implementan los servicios y están localizados en el plano de aplicación. Su función principal es proporcionar servicios de valor añadido dentro de la red IMS y a sus usuarios. Dependiendo del servicio proporcionado actúan como SIP-Proxies, UAs o B2BUA.

Los servidores de aplicación interactúan dentro de la red IMS con el S-CSCF para formar parte del control de la sesión y con el HSS para actualizar el perfil del usuario y sus cambios de estado.

Existen tres tipos de servidores de aplicación:

- SIP-AS: Es el servidor de aplicaciones originario que almacena y ejecuta servicios multimedia IP basados en SIP. Puede estar localizado en la red local o en la red visitada.

- OSA SCS (*Open Service Architectura Service Capability Server*). Es la pasarela entre IMS y OSA. Puede estar localizado en la red local o en la red visitada.

- IM-SSF (*IP Multimedia Service Switching Function)*. Es la pasarela entre IMS y CAMEL (*Customized Applications for Mobile network Enhaced Logic*). Localizado en la red local.

2.3.4 Principales protocolos utilizados en IMS

Dentro de IMS cabe mencionar los principales protocolos que emplea para su correcto funcionamiento. Todos los protocolos utilizados han sido desarrollados por el IETF, estos protocolos son SIP (RFC2543), SDP (RFC3264), DIAMETER (RFC3588) y RTP(RFC1889)/RTCP (RFC3550).

El protocolo SIP es utilizado junto con SDP en el plano de señalización para el establecimiento de sesión. El protocolo DIAMETER es utilizado para proporcionar las funcionalidades de Autenticación, Autorización y Facturación (AAA). Por último los protocolos RTP y RTCP son utilizados en el plano de datos para el transporte de información multimedia.

2.4 SIP

El protocolo SIP (*Session Initiation Protocol*) es un protocolo de señalización y control, desarrollado por el IETF, empleado para crear, modificar y finalizar sesiones multimedia. La primera especificación de SIP, RFC 2543, fue redactada en 1999, sustituyéndose en 2001 por la RFC 3261 [11].

SIP es un protocolo que permite establecer comunicaciones independientes del contenido de las mismas, pero no es un protocolo que proporciona de forma íntegra comunicaciones multimedia y por ello se utiliza en conjunto con otros protocolos:

- RTP (*Real-Time Transport Protocol*): Para el transporte de datos en tiempo real.

- SDP (*Session Description Protocol*): Proporciona la descripción de la sesión multimedia que se va a establecer.

- RSVP (*Reservation Protocol*): Reserva de recursos multimedia que proporcionan la calidad de servicio.

SIP adopta un modelo cliente/servidor con una lógica petición/respuesta. La estructura de sus mensajes se basa en HTTP (*HyperText Transfer Protocol*) y SMTP (*Simple Mail Transfer Protocol*), protocolos utilizados para la navegación web y el envío de correos electrónicos respectivamente.

Las características fundamentales del protocolo de señalización SIP son las siguientes:

- Resolución de direcciones, mapeo de llamadas y redirección de llamadas.

- Descubrimiento dinámico del destino.

- Administración de la sesión.

- Intercambio de información sobre el medio para permitir el establecimiento de sesión.

Las ventajas principales que ofrece SIP frente a otros protocolos de señalización del mismo tipo, como puede ser H.323 [38]son:

- Simplicidad: dado que usa sintaxis parecida a HTTP y SMTP los tiempos de desarrollo de aplicaciones son relativamente cortos ya que la reutilización de código es posible.

- Extensibilidad: basándose en HTTP y SMTP se han desarrollado un grupo importante de funcionalidades.

- Interoperabilidad: es un protocolo abierto y esto permite interoperabilidad entre plataformas de distintos fabricantes.

- Modularidad: es capaz de trabajar de forma independiente de otros protocolos.

- Integración: es capaz de integrarse perfectamente con el mundo web.

2.4.1 Elementos de SIP

2.4.1.1 Clientes

Los clientes finales son identificados por direcciones únicas denominadas URI-SIP (*Uniform Resources Identifier- SIP*). Una URI es una cadena corta de caracteres que identifica unívocamente un recurso. El formato que tiene las URI-SIP es el siguiente:

SIP:[userInfo]hostport[parametros]

- *userInfo*: contiene información sobre el usuario, seguida de una letra "@".

- *hostport*: contiene un nombre de dominio o una dirección IP. Adicionalmente, puede incluir un numero de puerto.

- Parámetros: contiene parámetros adicionales. Se indican tras el carácter ";".

Un usuario de SIP utiliza un UA (*User Agent*) para enviar y recibir mensajes SIP. Los UA son los puntos finales que están en contacto con los usuario, pueden ser hardware o software.

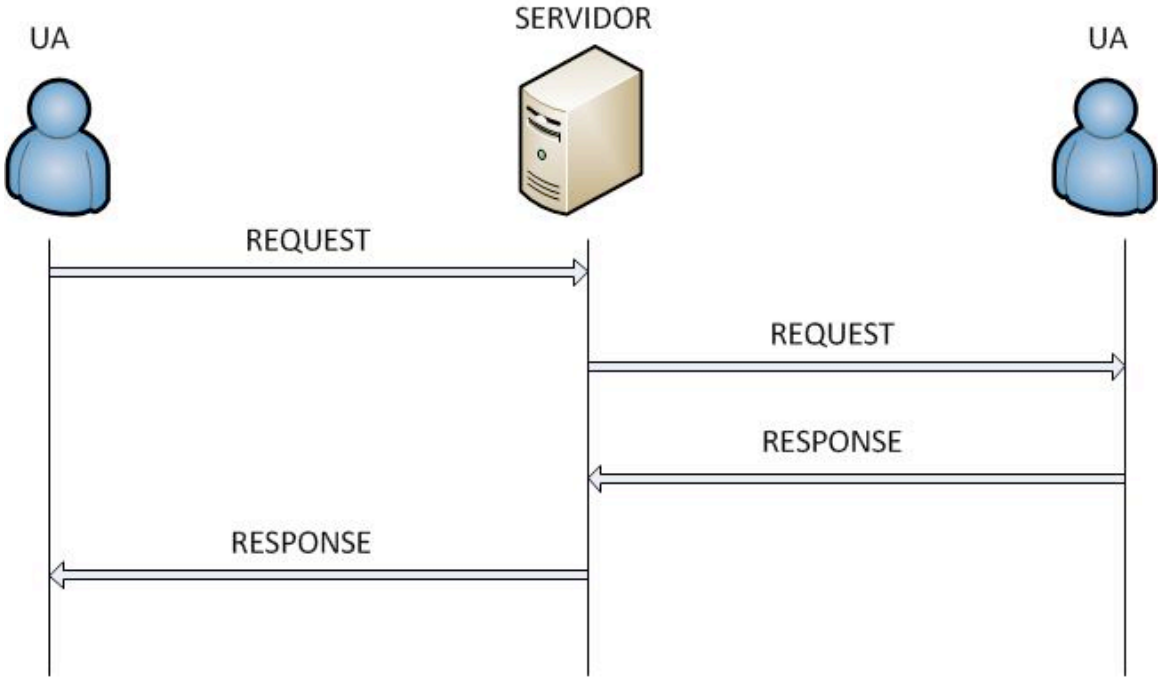


Figura 12: User Agent

2.4.1.2 Servidores

Los servidores se encargan de recibir peticiones de clientes o de otros servidores, procesarlas y dar una respuesta. Dentro de los servidores de SIP los tipos más destacados son:

- **Servidor de Registro**: Gestiona y almacena la información de localización de los usuarios. Mediante las peticiones de tipo REGISTER los usuarios indican su localización.

- **Servidor de Redirección**: Este tipo de servidor recibe peticiones de clientes o servidores SIP, obtiene la asociación entre la URI SIP del usuario al que se intenta contactar y sus direcciones de contacto actualizadas, y devuelve dichas direcciones de contacto al peticionario. Pueden requerir autenticación de los peticionarios y este tipo de servidor no encamina mensajes.

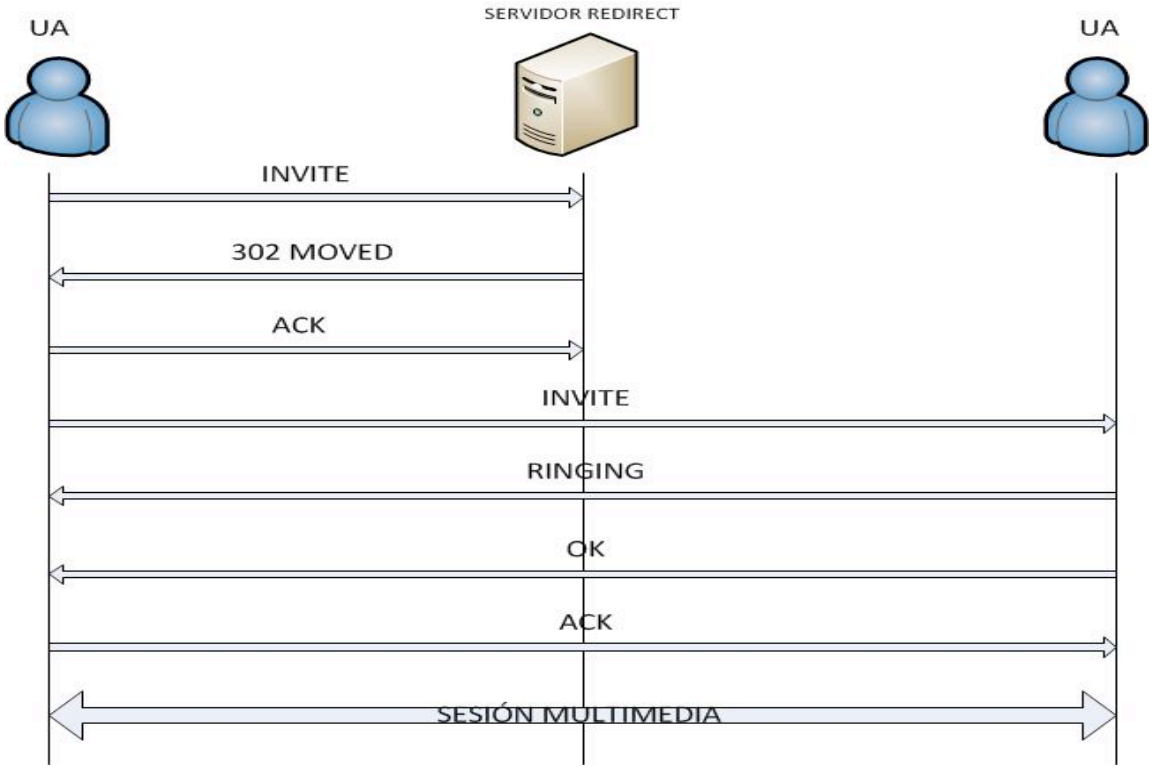


Figura 13: Servidor de redirección

- **Servidor Proxy**: Se comporta como entidad intermedia y puede actuar como cliente o como servidor, gestionando peticiones y respuestas. Cuando un cliente lanza una petición ésta podrá atravesar uno o varios poxies hasta llegar al destino. La respuesta a esa petición atravesará el mismo número de proxies. El servidor proxy puede proveer funciones tales como autenticación, autorización, control de acceso a la red, routing y seguridad.

- **Servidores B2BUA (Back To Back User Agent)**: Se comportan simultáneamente como clientes y servidores.

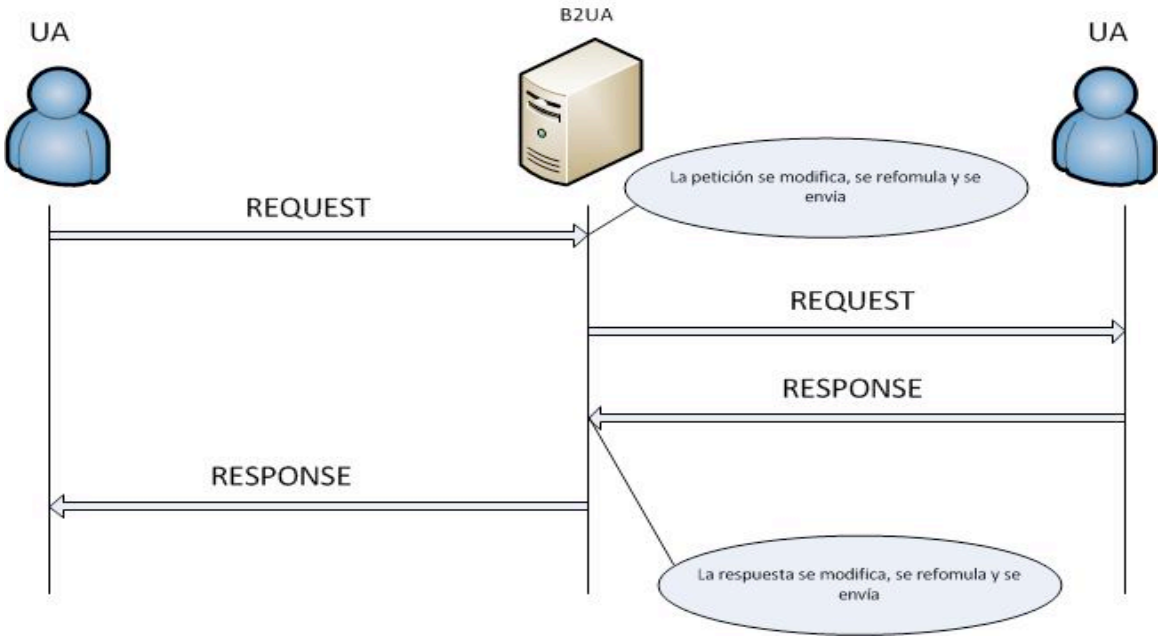


Figura 14: Servidor B2UA

- **Servidores Stateless y Statefull**: Los servidores de tipo Stateless no mantienen el estado a nivel de transacción, mientras que los de tipo Stafull si lo hacen.

- **Servidores Forking**: Servidor que es capaz de entregar las sesiones a varios UAs a la vez

2.4.2 Mensajes de SIP

SIP como protocolo de señalización va a usar mensajes para la conexión y control de las sesiones. Va a haber dos tipos de mensajes: las peticiones y las respuestas. Con estos dos tipos de mensajes se construyen las transacciones que darán lugar al establecimiento de sesiones entre dos clientes.

El formato, tanto de las peticiones como de las respuestas, es similar a HTTP y SMTP.

<start-line>
Campos de la cabecera
Línea en blanco
<cuerpo del mensaje>

- <start-line>: especifica el tipo de mensaje, es decir, índica si el mensaje es una petición o una respuesta.

- Campos de cabecera: tienen el formato *<nombre>:<valor>* .En todos los mensajes aparecen campos obligatorios y otros opcionales.

- <cuerpo del mensaje>: es opcional.

2.4.2.1 Peticiones

Las peticiones SIP son los mensajes emitidos por el extremo que inicia la comunicación. El campo <start-line> de las peticiones siempre tiene el siguiente formato:

<Método><Request-URI><Versión del protocolo>

En un principio SIP definió 6 métodos o peticiones principales en la RFC3261:

- **INVITE**: Utilizado para solicitar a un usuario el establecimiento de una sesión.
- **ACK**: Utilizado por el cliente que originó un INVITE para confirmar al servidor que le respondió que ha recibido la respuesta final al mismo
- **REGISTER**: Utilizado para informar de la localización de los usuarios
- **CANCEL**: Utilizado por los clientes para cancelar sesiones que aún estén pendientes de establecerse (el servidor aún no ha enviado una respuesta final); la recepción de un CANCEL por parte de un servidor aborta el intento de establecimiento de la sesión
- **BYE**: Utilizado por los usuarios para abandonar sesiones en curso. Es enviado por cualquier UA que esté participando en la sesión y en un mensaje extremo a extremo, *end-to-end*.
- **OPTIONS**: Utilizado para solicitar las capacidades de un servidor: métodos, protocolos y codecs soportados, etc.

Mas tarde en la RFC 3265 se aumentan los tipos de peticiones para dar mas funcionalidad y extensibilidad al protocolo.

- **REFER**: Utilizado para transferir llamadas.
- **SUSCRIBE**: Utilizado por los usuarios para solicitar la recepción de avisos cuando ocurran determinados eventos.
- **NOTIFY**: Utilizado para notificar el suceso de algún evento.
- **MESSAGE**: Utilizado para mensajería instantânea.
- **UPDATE**: Utilizado para modificar el estado de una sesión aún pendiente de establecer. Para las sesiones en curso la modificación debe realizarse mediante un nuevo INVITE.
- **INFO**: Utilizado para enviar información extremo a extremo entre usuarios que participan en una sesión; al ser extremo a extremo, ningún servidor SIP interpreta o actúa ante métodos de tipo INFO.

- **PRACK**: Utilizado para confirmar la recepción de respuestas provisionales, provisional ACK.

2.4.2.2 Respuestas

Las respuestas son mensajes emitidos por el receptor de la comunicación. Se identifican porque su campo <*start-line*> siempre presenta el siguiente formato:

<Versión del protocolo><*Status Code*><Resumen>

En la siguiente tabla se muestran los 6 tipos diferentes de respuestas que se pueden encontrar:

Rango	Tipo de respuesta	Comentarios
100-199	Informativa	De carácter provisional; indica el estado de la sesión antes de completarse; se interpreta como "*Request* recibido y en proceso"
200-299	Éxito	Definitiva; se interpreta como "*Request* recibido y procesado con éxito"
300-399	Redirección	Definitiva; utilizada para redirigir los *Requests* a otra dirección
400-499	Error en el cliente	Definitiva; el *Request* ha fallado por error en el cliente; se puede reintentar si se modifica como indica el *Response*
500-599	Error en el servidor	Definitiva; el *Request* ha fallado por error en el servidor; se puede reintentar en otro servidor
600-699	Fallo global	Definitiva; el *Request* no es valido para ningún servidor

Tabla 2: Códigos de respuesta a peticiones SIP

Algunas de las respuestas más comunes del protocolo son:

- **100 TRYING:** Como se puede ver pertenece al grupo de respuestas provisionales. Esta respuesta indica que la petición ha sido recibida por el siguiente salto y nunca es reenviada. Puede ser generada por un servidor proxy o por un UA.

- **180 RINGING:** Pertenece también al grupo de respuestas de carácter provisional. Indica que la petición INVITE ha sido recibida por el otro extremo y se está intentando alertar al usuario. Con esta respuesta también se indica al usuario que inicia la comunicación que al usuario llamado le está sonando el teléfono.

- **200 OK:** Pertenece al grupo de respuestas definitivas indicando que la petición ha sido procesada con éxito. Este tipo de respuesta incluye típicamente carga SDP, indica que se acepta la negociación iniciada en el INVITE.

2.4.2.3 Campos de la cabecera

En todas las peticiones y respuestas habrá campos que sean obligatorios y otros que sean opcionales. A continuación se describen los distintos campos, tanto obligatorios como opcionales, que se encuentran en la cabecera de las peticiones y respuestas:

- **TO**: Especifica el receptor lógico de la solicitud

  To: Bob <sip:bob@biloxi.com>;tag=1234

- **FROM**: Indica el usuario que origina la comunicación

  From: Alice <sip:alice@atlanta.com>;tag=5678

- **TAG**: Parámetro que sirve para la identificación de un diálogo. Se puede encontrar en los campos TO y FROM

- **Cseq**: Ayuda a unir peticiones y respuestas. Se incrementa cuando se transmite una nueva petición, excepto en el caso de que se transmita una petición de tipo ACK o CANCEL

  Cseq: 1 INVITE

- **Call-Id**: Identificador permanente de la comunicación

  Call-ID: a84b4c76e66710@pc33.atlanta.com

- **Max-Fordwards**: Usado para evitar bucles de encaminamiento. Se decrementa cada vez que el mensaje pasa por un proxy

  Max-Forwards: 70

- **VIA**: Es el campo que informa de todos los elementos intermedios que han participado en el proceso de establecimiento de la sesión. Junto con el campo Max-Fordwards evita que se formen bucles y principalmente es el campo que fuerza que las respuestas sigan el mismo camino que han seguido las peticiones. Debe aparecer en todos los mensajes, tanto en peticiones como en respuestas. En las peticiones cada proxy irá añadiendo su dirección o dominio y quitándolo en las respuestas.

  Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds

- **Route**: Fuerza el enrutamiento a las direcciones especificadas en dicho campo.

- **Contact**: Identifica la dirección en la que se quieren recibir los mensajes.

  Contact: <sip:alice@pc33.atlanta.com>

- **Allow**: Métodos permitidos en la comunicación

- **Content-Disposition**: Describe cómo debe ser interpretado el cuerpo del mensaje

- **Expires**: Indica el intervalo de tiempo durante el cual es válida la solicitud o contenido del mensaje

- **Content-Type**: Tipo de información que lleva el cuerpo del mensaje.

  Content-Type: application/sdp

- **Content-Length**: Tamaño del cuerpo del mensaje.

  Content-Length: 142

A continuación se puede ver ejemplo de petición y respuesta SIP que contienen los campos descritos para las peticiones y respuestas[11].

```
INVITE sip:bob@biloxi.com SIP/2.0
    Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
    Max-Forwards: 70
    To: Bob <sip:bob@biloxi.com>
    From: Alice <sip:alice@atlanta.com>;tag=1928301774
    Call-ID: a84b4c76e66710@pc33.atlanta.com
    CSeq: 314159 INVITE
    Contact: <sip:alice@pc33.atlanta.com>
    Content-Type: application/sdp
    Content-Length: 142
```

```
SIP/2.0 200 OK
      Via: SIP/2.0/UDP server10.biloxi.com
         ;branch=z9hG4bKnashds8;received=192.0.2.3
      Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
         ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
      Via: SIP/2.0/UDP pc33.atlanta.com
         ;branch=z9hG4bK776asdhds ;received=192.0.2.1
      To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
      From: Alice <sip:alice@atlanta.com>;tag=1928301774
      Call-ID: a84b4c76e66710@pc33.atlanta.com
      CSeq: 314159 INVITE
      Contact: <sip:bob@192.0.2.4>
      Content-Type: application/sdp
      Content-Length: 131
```

2.4.3 Registro, establecimiento y liberación

En los siguientes apartados se muestra el intercambio de mensajes SIP para el registro de usuarios, el establecimiento de sesión y la liberación o finalización de la sesión.

2.4.3.1 Registro

En la siguiente figura se puede ver el proceso de registro en SIP. Este proceso es obligatorio para todos los usuarios para poder mantener una sesión SIP. Mediante el método REGISTER se asocia y registra la dirección lógica del usuario con la dirección física en la que se encuentra el usuario.

El proceso de registro consta de dos pasos:

1. El usuario que quiere registrarse envía un mensaje de tipo REGISTER al servidor.

2. El servidor contesta con un OK al usuario y en ese momento el usuario queda registrado.

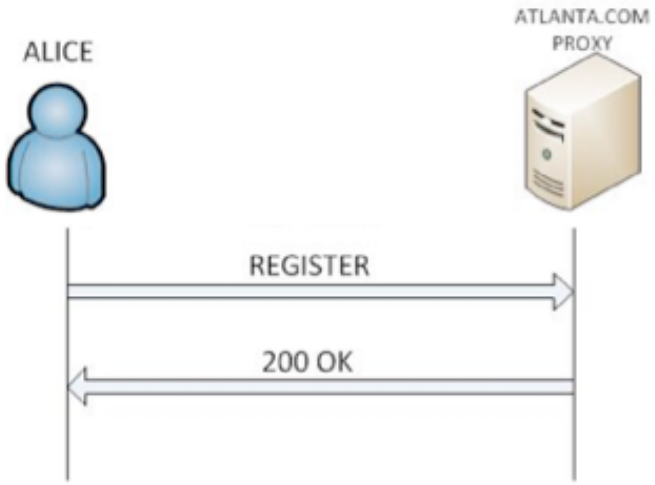


Figura 15: Proceso de registro

2.4.3.2 Establecimiento de sesión

En la siguiente figura se puede ver cuáles son los mensajes SIP que se intercambian para el establecimiento de sesión entre dos usuarios. Como se puede ver el usuario "Alice" quiere establecer una sesión con el usuario "Bob". Para ello en primer lugar el usuario Alice genera la correspondiente petición INVITE que envía a su servidor proxy. El servidor al recibir la petición será el encargado de reenviarla hasta el destino oportuno, para ello cuando recibe la petición genera una respuestas del tipo provisional, 100 Trying, para el usuario Alice que indica que la petición INVITE recibida está siendo tratada. La

petición recibida es reenviada al siguiente servidor que sigue la misma operativa que el primer servidor hasta que la petición es entregada al usuario final.

Una vez que la petición ha llegado al usuario final se genera otra respuesta de carácter provisional, 180 RINGING, que indica que la petición ha llegado a su destino y está siendo tratada por el usuario final.

Una vez el usuario final ha tratado la petición envía una respuesta definitiva, 200 OK, al usuario que generó la petición para iniciar la comunicación.

Una vez se reciba esta respuesta definitiva por parte del primer usuario, éste generará el mensaje de asentimiento ACK y empezará a transmitirse e intercambiarse el contenido multimedia, ya que la sesión ya ha sido establecida.

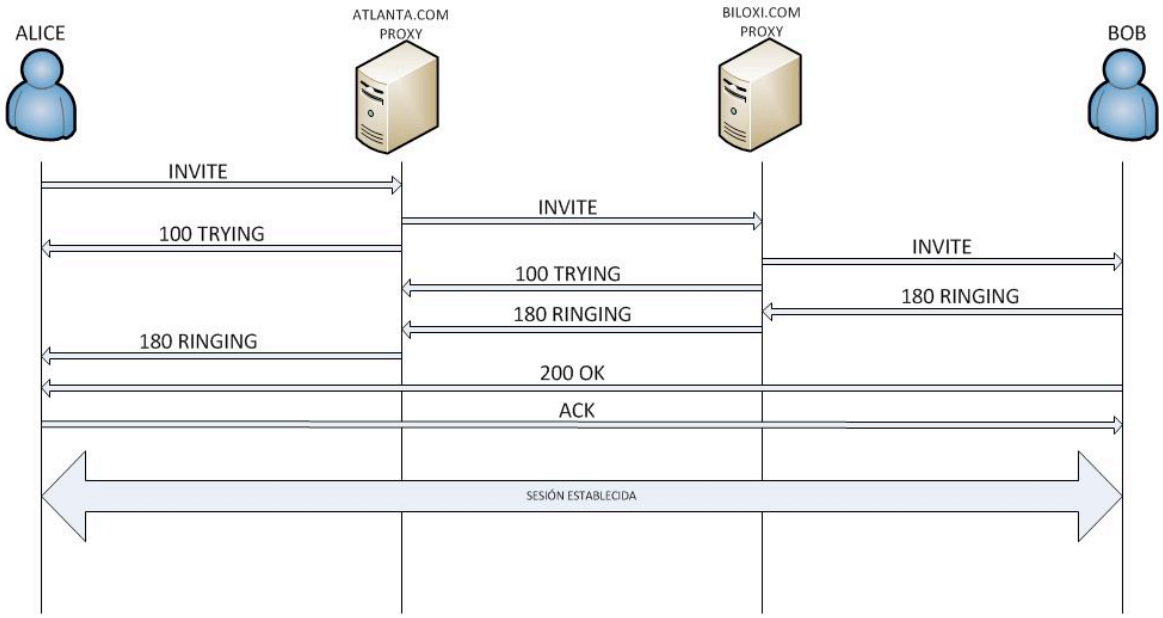


Figura 16: Establecimiento de sesión

2.4.3.3 Liberación de sesión

Una vez que la sesión ha sido establecida, cualquiera de los dos extremos de la comunicación puede liberarla. Para liberar la sesión cualquiera de los extremos puede generar una petición de tipo BYE. A diferencia del INVITE este tipo de petición no pasa por los servidores proxys, es una petición extremo a extremo.

El usuario que recibe la petición de liberación de la sesión responde con una respuesta definitiva, 200 OK, y en ese momento la sesión que había establecida entre los do UAs queda liberada.

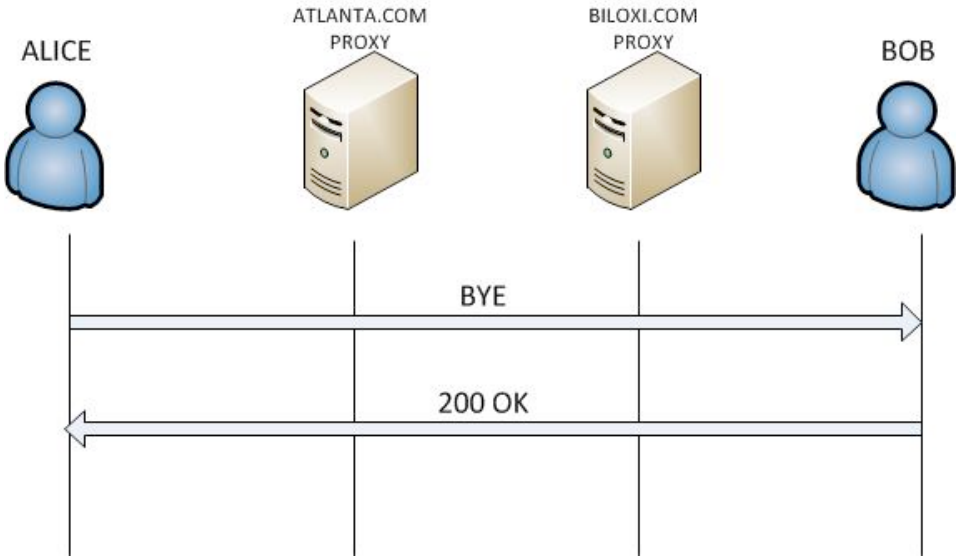


Figura 17: Liberación de la sesión

2.5 SDP

El protocolo SDP (*Session Description Protocol*) fue desarrollado por el grupo MMUSIC (*Multiparty Multimedia SessIon Control*) del IETF en el RFC3264 [12]. Es un protocolo que ofrece la posibilidad de negociar los parámetros de una sesión mediante un modelo oferta/respuesta en el cual cada una de las dos entidades puede rechazar codecs e incluso descripciones multimedia. De esta manera se pueden ir negociando los parámetros de la sesión hasta establecer un acuerdo por ambas partes. Entre otras cosas, SDP permite la negociación de los parámetros de calidad del servicio para establecer una sesión multimedia, así como el tipo de flujo que se va a transmitir.

La carga del mensaje SDP se encapsula dentro del cuerpo de las peticiones y respuestas del protocolo SIP. Típicamente irá encapsulada en los mensajes INVITE, donde el emisor de la comunicación lanzará una primera oferta con los codecs que su equipo es capaz de manejar, y en los mensajes de tipo OK, donde el receptor de la comunicación responderá con los codecs que tienen en común emisor y receptor.

Una sesión se describe con un conjunto de atributos, cada uno de ellos en una línea. Cada atributo está abreviado con una letra minúscula indicando el tipo de campo que representa seguido de su valor. Dentro de SDP se pueden encontrar 3 grandes grupos de atributos:

- **Atributos de descripción de la sesión**: Contienen información referente a la sesión, tal como, versión del protocolo, origen….

- **Atributos de descripción de tiempos**: Contiene información sobre el tiempo de inicio y finalización de la sesión, así como el número de repeticiones.

- **Atributos de descripción multimedia**: Contiene información referente al protocolo de transporte e información de la conexión.

En la siguiente tabla se muestran los atributos del protocolo con una breve descripción:

Atributos de descripción de sesión	
Id	**Descripción**
v	Versión del protocolo.
o	Origen e identificador de sesión
s	Nombre de sesión
i*	Información de la sesión
u*	URI de descripción
e*	Correo electrónico
p*	Número telefónico
c	Información de conexión
b*	Información de ancho de banda
z*	Ajustes de zona horaria
k*	Clave de cifrado
a*	Atributos de sesión
Atributos de descripción de tiempos	
Id	**Descripción**
t	Tiempo durante el cual la sesión estará activa
r*	Repetición
Atributos de descripción multimedia	
Id	**Descripción**
m	Nombre de medio y dirección de transporte
i*	Título
c*	Información de conexión
b*	Información de ancho de banda
k*	Clave de cifrado
a*	Atributos de sesión
* Indica los parámetros opcionales	

Tabla 3: Atributos de SDP

Un ejemplo de sesión SDP se puede ver a continuación (RFC 4566)

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

2.5.1 Atributos obligatorios en sesiones SDP

En este apartado se hace una descripción de los atributos obligatorios que todas las sesiones SDP deben contener. En el anexo B se encuentra una descripción de todos los atributos opcionales del protocolo.

2.5.1.1 Protocol version ("v=")

Este campo indica la versión del protocolo SDP. En este momento el único número de versión existente es el 0, por tanto el valor de este campo será "v=0".

2.5.1.2 Origin ("o=")

El campo origen se utiliza para identificar de manera unívoca la sesión SDP. El formato de este campo, que da información sobre el origen de la sesión SDP, es el siguiente:

o = <username> <sess-id> <sess-version> <nettype> <addrtype> <unicast-address>

- **username**: nombre del usuario que genera la sesión.
- **sess-id**: identificador único de la sesión formado por una cadena numérica.
- **sess-version**: número que identifica la versión de la sesión.
- **nettype**: cadena de texto que identifica el tipo de red que se está utilizando. Su valor es siempre "IN" para Internet.
- **addrtype**: cadena de texto que indica que tipo de dirección se está utilizando. El valor que puede adoptar este campo es IP4 ,para direcciones del tipo IPv4, e IP6, para direcciones del tipo IPv6.
- **unicast-address**: dirección desde la cual ha sido generada la sesión SDP. Puede ser de tipo IPv4 o IPv6.

En general el campo "o=" sirve como identificador general de la sesión independientemente de los cambio que puedan darse.

2.5.1.3 Session name ("s=")

Este campo está formado por una cadena de texto que indica el nombre de la sesión. Sólo puede haber un campo de este tipo por cada sesión.

s=<session name>

2.5.1.4 Connection data (“c=”)

Este campo contiene información de la conexión y tiene el siguiente formato:

c= <nettype> <addrtype> <connection-address>

- **nettype**: cadena de texto que identifica el tipo de red que se está utilizando. Su valor es siempre “IN” para internet.

- **addrtype**: cadena de texto que indica que tipo de dirección se está utilizando. El que puede adoptar este campo es IP4 ,para direcciones del tipo IPv4, e IP6, para direcciones del tipo IPv6.

- **connection-address**: indica la dirección de conexión. Para sessiones multicast indica la dirección IP del grupo multicast y el puerto destino. Para sesiones unicast indica la dirección IP y el puerto al que se indica la información multimedia.

2.5.1.5 Timing (“t=”)

El campo Timing indica el inicio y finalización de una sesión.

t = <start-time> <stop-time>

Dentro de una sesión SDP puede haber múltiples líneas de este campo si la sesión esta activa en intervalos irregulares de tiempo.

2.5.1.6 Media description (“m=”)

El campo media indica qué tipo de información multimedia será intercambiada en la sesión. En una sesión SDP podemos encontrar uno o mas campos de este tipo. El formato que tiene este campo es el siguiente:

m= <media> <port> <proto> <fmt>

- **media:** indica el tipo de datos multimedia que se va a intercambiar en la sesión. Estos datos pueden ser audio, vídeo, texto, imágenes, aplicaciones o mensajes.

- **port:** indica el número de puerto por el que se envía el flujo multimedia.

- **proto:** indica el protocolo de transporte sobre el cual se está transmitiendo la sesión.

- **fmt:** muestra los formatos en los que el flujo multimedia puede ser codificado. Se representa mediante una cadena numérica en la que los números van espaciados entre si y cada número identifica un formato de codificación .

2.6 RTP

RTP (*Real time Transport Protocol*) fue desarrollado por el grupo de trabajo AVT (*Audio/Video Transport*) del IETF en la RFC 1889 y más tarde actualizado en la RFC 3550 [13]. Es un protocolo de nivel de transporte utilizado para la transmisión de información en tiempo real, como por ejemplo audio y vídeo en una videoconferencia.

El protocolo RTP define realmente dos protocolos:

- RTP (Real Time Transport Protocol): Utilizado para transportar los datos multimedia como la voz y el vídeo. Se encarga de la detección de pérdidas y el etiquetado de contenidos.

- RTCP (Real Time Transport Control Protocol): Utilizado para enviar periódicamente información de control asociada con el flujo de datos. Controla la calidad de servicio.

2.6.1 RTP

El protocolo RTP se establece en el espacio de usuario y se ejecuta, por lo general, sobre UDP, ya que posee menor retardo que TCP. Por tanto con UDP se gana velocidad a cambio de sacrificar la confiabilidad que ofrece TCP. RTP no garantiza la entrega de todos los paquetes, ni la llegada de éstos en el instante adecuado.

La función básica de RTP es multiplexar varios flujos de datos en tiempo real en un solo flujo de paquetes UDP, pudiéndose enviar tanto a un solo destino (unicast) o múltiples destinos (multicast). Los paquetes son numerados de la siguiente manera: se le asigna a cada paquete un número mayor que su antecesor. Esto será útil para que la aplicación conozca si ha fallado algún paquete en la transmisión. Si ha fallado, al no tener un control de flujo, de errores, de confirmaciones de recepción ni de solicitud de transmisión, la mejor opción es la interpolación de los datos.

Otra característica muy importante para las aplicaciones de contenido multimedia en tiempo real es el time-stamping (marcación del tiempo). La idea es permitir que el origen asocie una marca de tiempo con la primera muestra de cada paquete. Las marcas de tiempo son relativas al inicio del flujo, por tanto, solo importa las diferencias entre dichas marcas de tiempo. Con este planteamiento, el destino es capaz de almacenar un pequeño buffer e ir reproduciendo cada muestra el número exacto de milisegundos después del inicio del flujo reduciendo los efectos de retardo y sincronizando.

2.6.2 RTCP

El protocolo RTCP es complementario a RTP, ofreciendo un control de la calidad del servicio. Utiliza UDP por el puerto siguiente al puerto que se utiliza para RTP. El protocolo RTCP se basa en la transmisión periódica de paquetes de control ofreciendo

información sobre la calidad de los datos distribuidos por la fuente. Por tanto, la función principal de RTCP es la de proveer una realimentación de la calidad de servicio.

2.7 ICE

ICE (*Interactive Connectivity Establishment)* fue desarrollado por el grupo de trabajo MMUSIC (*Multiparty Multimedia Session Control*) del IETF en la RFC5245 [26] en abril de 2010.

ICE define un protocolo de actuación gracias al cual dos dispositivos SIP son capaces de mantener una sesión multimedia salvando todas las dificultades que el NAT (*Network Address Translation*) pueda poner de por medio. ICE permite que los dispositivos involucrados en la sesión SIP prueben distintos medios o rutas para comunicarse entre sí y acuerden uno común.

El funcionamiento detallado del ICE se puede dividir en seis etapas: recolección, priorización, codificación, oferta y respuesta, verificación, y completado. Se puede ver un resumen de estas seis etapas en la siguiente tabla.

Etapa	Descripción
Recolección	Reunión de posibles candidatos con dirección IP y puerto
Priorización	Asignación de prioridad a todos los candidatos reunidos
Codificación	Codificación de candidatos dentro de la petición SDP correspondiente
Oferta/Respuesta	Intercambio de sesiones SDP entre las dos partes de la comunicación
Verificación	Se realiza una relación de los candidatos de ambos extremos
Completado	Se encuentra el par adecuado para realizar la comunicación

Tabla 4: Etapas del protocolo ICE

2.8 STUN

STUN (*Simple Transversal Utilities for NAT*) fue desarrollado por el IETF en marzo de 2003 en la RFC3489 y actualizado en Octubre de 2008 en la RFC5389 [27].

STUN es un protocolo de tipo cliente-servidor, el cliente envía una petición al servidor y el servidor devuelve una respuesta. Dichas peticiones se utilizan para determinar las asociaciones IP-Puerto privado/IP-Puerto público establecidas por los NATs.

El cliente envía una petición de asociación al servidor sobre UDP, el servidor examina la IP y el puerto origen y los copia en la respuesta que envía al cliente. Hay varios parámetros en la petición que permiten al cliente pedir que la respuesta se envía a otra dirección o que el servidor envía la respuesta desde otra IP o puerto diferentes. STUN se utiliza para descubrir la presencia de NAT, y para aprender las asociaciones que

dicho NAT establece. El cliente STUN suele estar dentro en una aplicación que necesita obtener una dirección y un puerto públicos para recibir datos.

La petición de STUN se utiliza para descubrir la presencia de NAT, y para descubrir la IP y el puerto público que se corresponden en dicho NAT con la IP y el puerto privado. Las peticiones son enviadas utilizando UDP, cuando una petición llega al servidor este copia la IP y puerto origen en la respuesta STUN, para todos los tipos de NAT posibles esta respuesta llega al cliente.

2.9 Lenguajes de programación

2.9.1 Java

Java [14] es un lenguaje de programación orientado a objetos que fue desarrollado por Sun Microsystem en la deácada de los 90. En un principio java fue pensado para utilizarse en cualquier tipo de electrodoméstico. Se pretendía crear un hardware polivalente con un sistema opetativo eficiente (*SUN Operative system SunOs*) y un lenguaje de desarrollo denominado Oak, pero el proyecto fracasó.

Fue en 1995 con el auge y fácil acceso a Internet cuando sun decidió retomar el proyecto de creación de un lenguaje y le dio el nombre de Java. Gracias a la decisión de distribuirlo libremente a través de la red el lenguaje se popularizó y consiguió un gran número de programadores que lo depurasen. A partir de este momento el lenguaje se extiende a una gran velocidad y se crean numerosas clases para el manejo de la pila de protocolos de TCP/IP. Es entonces, en 1995, cuando sun lanzá las primeras versiones de Java.

La principal caraterística de java es que es un lenguaje compilado e interpretado. Todo programa de java ha de compilarse y el código que se genera es interpretado por una máquina virtual JVM (*Java Virtual Machine*). De esta forma se consigue un lenguaje independiente de la arquitectura y plataforma de ejecución, ya que el código se ejecuta en la máquina virtual. El uso de la JVM permite portar con poco esfuerzo a cualquier procesador o sistema operativo cualquier aplicación desarrollada en este lenguaje.

La ejecución del código Java es segura y fiable. Los programas no acceden directamente a la memoria del ordenador, siendo imposible que un programa escrito en Java pueda acceder a los recursos del ordenador sin que esta operación le sea permitida de forma explícita.

Con el lenguaje java se pueden hacer varios tipos de programas:

- **Applets**: pequeños programas que se incorporan en una página web y que por tanto, necesitan un navegador web compatible con java para ejecutarse. A menudo los applets se descargan junto con una página HTML desde un servidor web y se ejecutan en el cliente.

- **Aplicaciones**: programas de propósito general que normalmente se ejecutan desde la línea de comandos del sistema operativo.

- **Servlets**: programas que están pensados para trabajar en el lado del servidor y desarrollar aplicaciones web que interactúen con clientes. Se verán con mas detalle en al apartado 2.10.

2.9.2 JavaScript

JavaScript [15] [16] es un lenguaje desarrollado por NetScape y Sun para simplificar la creación de contenidos interactivos en páginas web sin necesidad de crear applets en Java.

JavaScript es un lenguaje interpretado, es decir, no se compila de forma previa a la ejecución. El código fuente se incorpora directamente dentro de la página web en el HTML y el navegador es el encargado de interpretar el código. Por este motivo, a diferencia de java, las comprobaciones de validez de referencias a objetos se realizan en tiempo de ejecución.

JavaScript fue diseñado para añadir dinamismo e interactividad a las páginas HTML, ya que es un lenguaje que responde a eventos. Los eventos a los que responde este lenguaje, cuándo ocurren y el manejador que llevan asociado lo podemos ver el la siguiente tabla

Eventos	**Ocurre cuando…..**	**Manejador**
load	Se carga la página del navegador	onLoad
unload	Se descarga la página del navegador	onUnload
click	El usuario hace click sobre un elemento	onClick
mouseover	El usuario mueve el ratón sobre una referencia	onMouseOver
mouseout	El usuario mueve el ratón fuera de una referencia	onMouseOut
focus	El usuario entra en un campo de un formulario	onFocus
blur	El usuario sale de un campo de un formulario	onBlur
change	El usuario cambia el valor de un campo de texto	onChange
select	El usuario selecciona un elemento de un campo de selección	onSelect
submit	El usuario envía un formulario	onSubmit

Tabla 5: Eventos de JavaScript

2.9.3 HTML

HTML [17] son las siglas de *HyperText Markup Language*. HTML no es un lenguaje de programación, es un lenguaje de especificación de contenidos. Mediante HTML se puede especificar, usando un conjunto de marcas, cómo va a representarse la información en un navegador.

HTML es un estándar a cargo del W3C. En noviembre de 1995 se presenta la primera versión oficial de HTML denominada HTML 2.0 y presentada en la RFC 1866.

Desde entonces HTML ha seguido una continua evolución hasta el día de hoy. Hoy en día la última versión de HTML que está en desarrollo es HTML 5.

Aunque está en fase de desarrollo HTML 5 [18] ya es usado por multitud de desarrolladores por las novedades y beneficios que incluye respecto a las anteriores versiones de HTML. Las mejoras mas destacables que presenta respecto a versiones anteriores son:

- Simplificación: Ofrece formas mas sencillas de especificar algunos parámetros.

- Contenido multimedia: Dos de las novedades más importantes son la introducción de las etiquetas <audio> y <video>. Por un lado la etiqueta <video> permite insertar vídeos en la página web sin necesidad de plug-ins. Cada navegador soporta codecs diferentes de vídeo, lo que hace necesario recodificar el vídeo en múltiples formatos. Por otro lado la etiqueta <audio> permite insertar pistas de audio sin necesidad de plug-ins y al igual que la etiqueta <video> obliga a codificar estas pistas de audio en varios formatos.

- Almacenamiento de datos del lado del cliente.

- Nueva versión de hojas de estilo CSS.

- Geo-localización: Los sitios web podrán saber la localización física de la persona que los visita.

2.9.4 XML

XML [19] son las siglas de *eXtensible Markup Language*. XML es un meta-lenguaje, creado por el W3C, que permite definir lenguajes de marcado adecuados para usos determinados. XML es una adaptación de SGML(*Standard Generalized Markup Language*) simplificado y adaptado a Internet.

XML representa información estructurada en la web, de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa , por una gran variedad de dispositivos y tipos de aplicaciones. Entre las características más destacables de XML es que fácilmente procesable y separa radicalmente el contenido y el formato de presentación.

2.10 Servlets

Los Servlets [20] son programas, escritos en lenguaje Java, que se ejecutan en el servidor, realizando la función de una capa intermedia entre una petición que proviene de un navegador Web u otro cliente HTTP, y las aplicaciones del servidor. Su función

principal es proveer páginas web dinámicas y personalizadas, utilizando para este objetivo el acceso a bases de datos y otros recursos.

Para poder utilizar servlets dentro de aplicaciones web, se debe tener acceso a un contenedor de Servlets, que no es mas que un servidor que tenga soporte para el API de Servlets.

El API Servlet propone una jerarquía de clases bien definidas para el trabajo con los mismos, donde se puede encontrar desde servlets sin implementaciones definidas, como la clase *GenericServlet* donde se debe realizar todo el trabajo hasta la clase *HttpServlet* donde sólo se implementarán los métodos de acceso *doGet()* y *doPost()*. El principal componente del API es la interfaz Servlet. Este interfaz contiene los principales métodos para manipular, no sólo los servlets, si no también las comunicaciones con los clientes. Todos los Servlets deben implementar esta interfaz ya sea directa o indirectamente. En la siguiente figura se puede ver la jerarquía de los servlets.

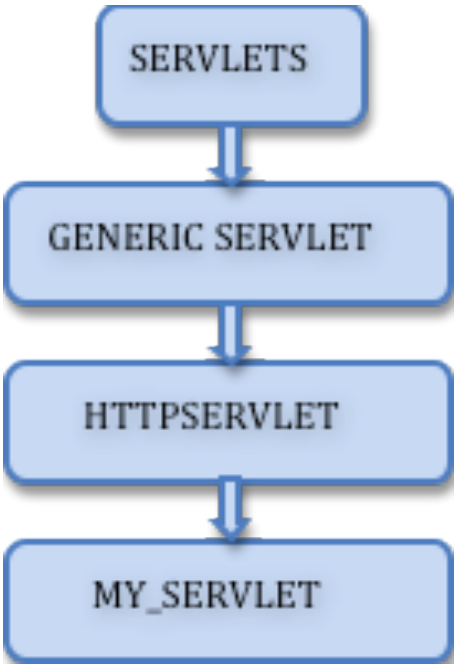


Figura 18: Jerarquía Servlets

Los Servlets Java tienen un ciclo de vida marcado por tres métodos de la interfaz Servlet: *init(), service() y destroy()*. El método *init()* lo invoca el contenedor de Servlets, pata inicializar el servlet. El método *service()* realiza la gestión de peticiones y respuestas y el método *destroy()* se invoca cuando el servlet va a ser descargado del servidor y los recursos quedan liberados:

Figura 19: Ciclo de vida Servlet

2.11 Apache-Tomcat

Apache Tomcat [21] es un servidor web multiplataforma que funciona como contenedor de servlets y que fue desarrollado por el *Apache Software Fooundation*(ASF). Implementa las especificaciones de los Servlets y de las *Java Server Pages* (JSP). Como ya se ha dicho en 2.8 los servlets son programas escritos en lenguaje java que se ejecutan en el servidor y los JSP son páginas dinámicas que se ejecutan en el servidor.

Dentro de apache-tomcat cabe destacar su jerarquía de directorios:

- **bin**: contiene ejecutables y scrips de tomcat
- **conf**: contiene la configuración global para todas las aplicaciones
- **lib**: contiene ficheros JAR disponibles para todas las aplicaciones web
- **logs**: contiene ficheros de log
- **webapps**: es el directorio base para aplicaciones web
- **work**: contiene servlets y clases resultantes de traducir ficheros JSP
- **temp**: contiene archivos temporales

2.12 Jain-SIP

Jain-Sip [22] es un conjunto de APIs de JAVA para el manejo del protocolo SIP de una forma estandarizada e interoperable. Antes de la existencia de Jain-SIP cada desarrollador de aplicaciones implementaba el protocolo de una forma no estandarizada y esto impedía la interoperabilidad de aplicaciones de distintos fabricantes y desarrolladores.

Jain-Sip se basa en el manejo del protocolo SIP a través de una pila y sobre esta pila se garantiza la interoperabilidad de aplicaciones.

La especificación de Jain-SIP define:

- La interfaz de la pila
- La interfaz de los mensajes

- Los eventos

Estas APIs son utilizadas para el desarrollo y la implementación del protocolo SIP. La implementación está basada en un modelo asíncrono de eventos del tipo *Listeners/Providers*. Este tipo de implementación proporciona a la aplicación la capacidad de tratar y procesar el envío y recepción de mensajes del protocolo SIP. En este modelo se utilizan identificadores para agrupar los mensajes en dos grupos: Diálogos y Transacciones.

Una transacción está formada por dos mensajes, una petición y la respuesta correspondiente, mientras que un diálogo está formado por varios mensajes, un diálogo puede contener varias transacciones.

En el siguiente apartado se verá la arquitectura de Jain-Sip y como interaccionan cada uno de sus elementos.

2.12.1 Arquitectura de objetos Jain-SIP

La arquitectura de Jain-Sip está compuesta por los siguientes elementos

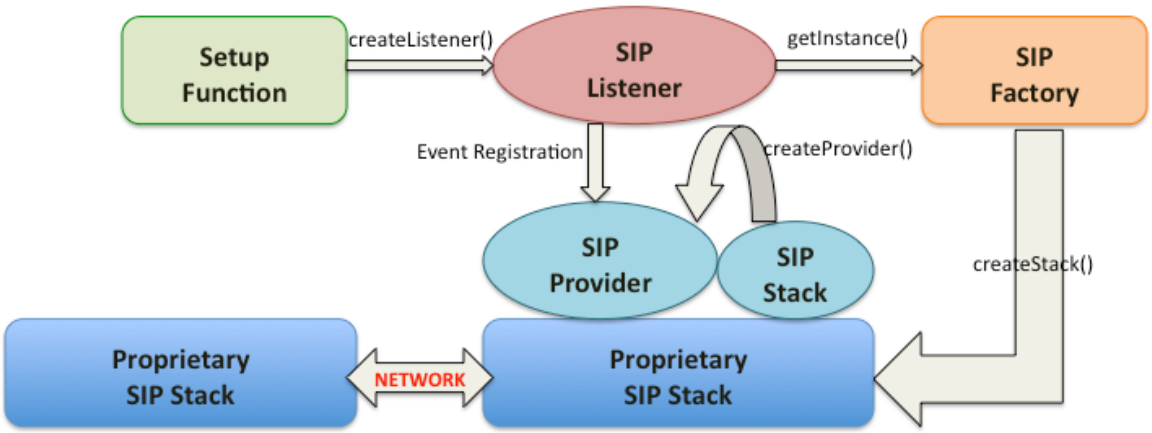


Figura 20: Arquitectura Jain-SIP

- Interfaz SIPListener: es la encargada de gestionar los eventos que ocurren cuando se recibe o envía una petición.

- Interfaz SIPProvider: Es la encargada de generar y enviar las peticiones SIP y de recibir y procesar las respuestas recibidas. Es la entidad que maneja los mensajes de la pila SIP.

- Interfaz SIPStack: define la pila SIP de la aplicación, a través de la cual se podrán generar las peticiones y recibir las respuestas. Esta pila tiene que tener las siguientes propiedades: Debe tener una dirección IP asignada, un nombre como identificador, un proxy de salida y un filtro de retransmisión.

Cada una de estas interfaces se engloba dentro de un ámbito diferente dentro del protocolo, esto queda reflejado en la siguiente figura:

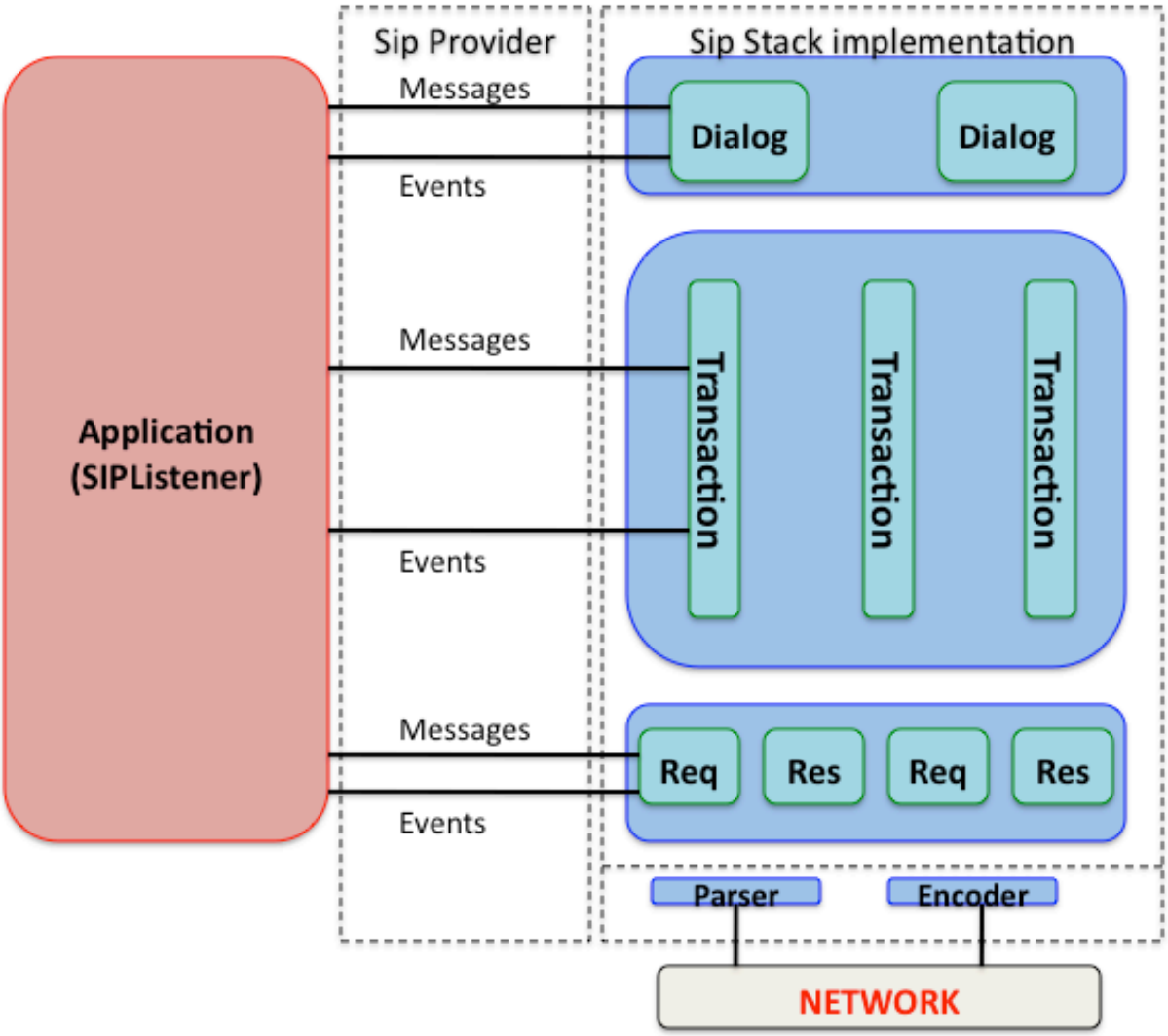


Figura 21: Estructura de la implementación de Jain SIP

2.12.2 Arquitectura de mensajes Jain-SIP

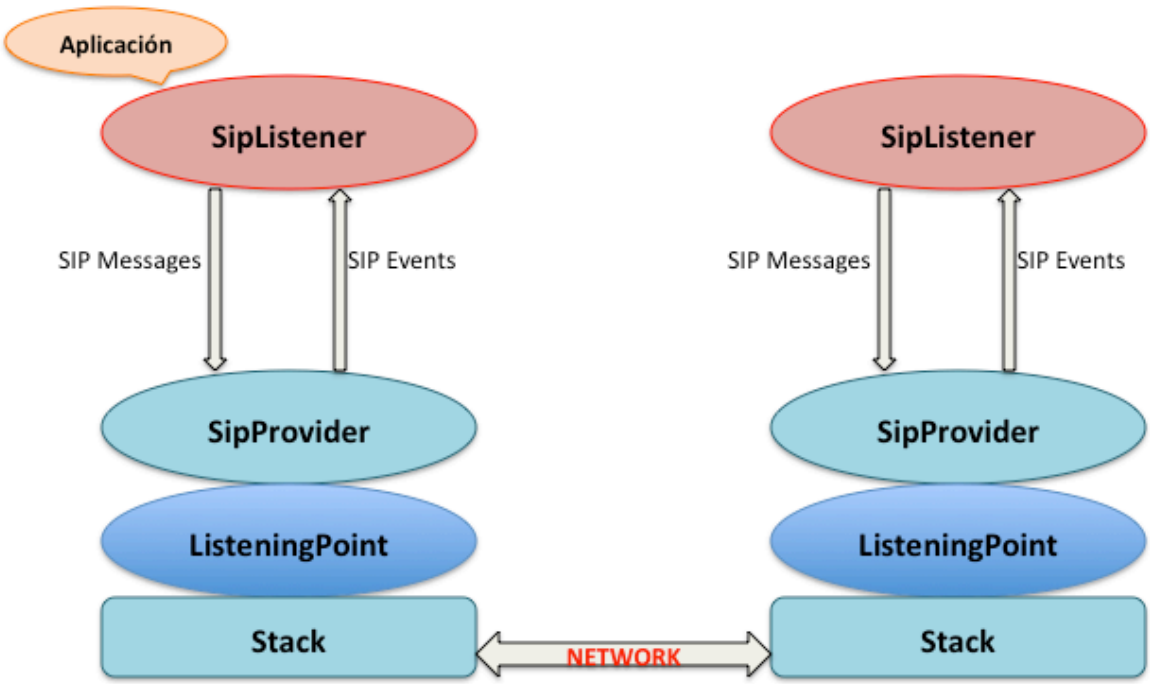


Figura 22: Arquitectura de mensajes Jain-SIP

Como se ha dicho anteriormente, esta arquitectura está basada en un modelo asíncrono de eventos Listeners/Providers. Existe una relación directa entre un evento de un proveedor y un evento de un "oyente".

Los eventos son los encargados de encapsular tanto las peticiones (*Request*), mensajes que van del ciente al servidor, como las respuestas (*Response*) mensajes que van del servidor al cliente.

SipListener representa los eventos usuario, de forma que se queda a la escucha (entrada de eventos) de mensajes, que pueden pertenecer a un diálogo ya establecido, o pueden referirse a un nuevo diálogo.

SipProvider representa el elemento proveedor que se encarga de recibir los mensajes desde la red y se los envía al nivel de aplicación como eventos.

A parte de estas interfaces descritas anteriormente, Jain-SIP describe 4 factorías para el maneje de las peticiones y respuestas del protocolo SIP:

- SipFactory: Define los métodos para crear la pila SIP y objetos de otras factorías.
- AddressFactory: Define los métodos para crear las direcciones SIP, URIs SIP.

- HeaderFactory: Define los métodos para crear las distintas cabeceras de los mensajes SIP.
- MessageFactory: Define los métodos para crear nuevas peticiones y respuestas.

2.13 Ajax

AJAX es el acrónimo de *Asynchronous JavaScript And Xml*. El término AJAX apareció por primera vez en el artículo "AJAX: A New Approach to Web Applications" publicado por Jesse James Garrett el 18 de febrero de 2005 [23].

En dicho artículo se describe AJAX de la siguiente forma:

"*Ajax isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways.*"

Las tecnologías que forman AJAX son:

- HTML y CSS para crear una presentación basada en estándares
- DOM para la inserción y manipulación dinámica de la presentación
- XML, XSLT y JSON para el intercambio y manipulación de la información
- XMLHttpRequest para el intercambio asíncrono de información
- JavaScript para unir todas las tecnologías nombradas

En la siguiente figura se puede ver cómo se interconectan todas las tecnologías:

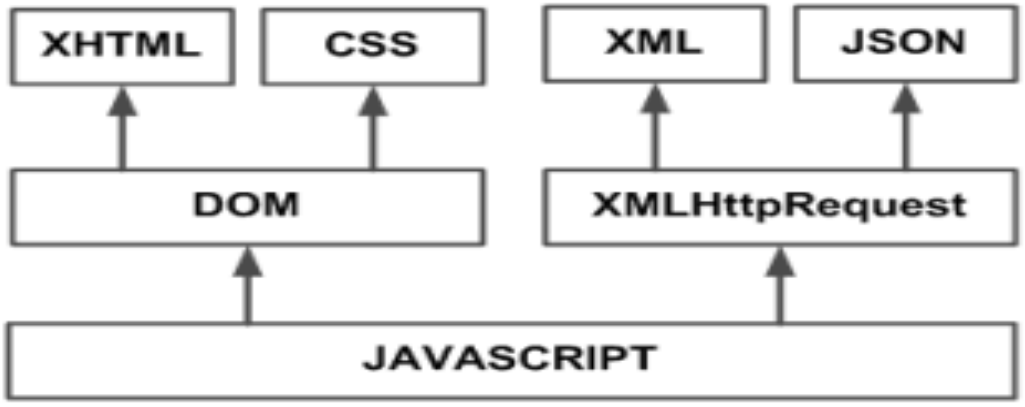


Figura 23: Interconexión de tecnologías en AJAX

En las aplicaciones web tradicionales cualquier acción ejecutada por el usuario desencadena en una llamada al servidor y una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador. Esta técnica tradicional presenta el inconveniente de que al generar peticiones continuas al servidor el usuario debe esperar a que se recargue la página con los cambios solicitados. Este problema viene solucionado en AJAX por lo que se denomina "AJAX-Engine". En la siguiente figura se puede ver el modelo tradicional de las aplicaciones web y el modelo AJAX:

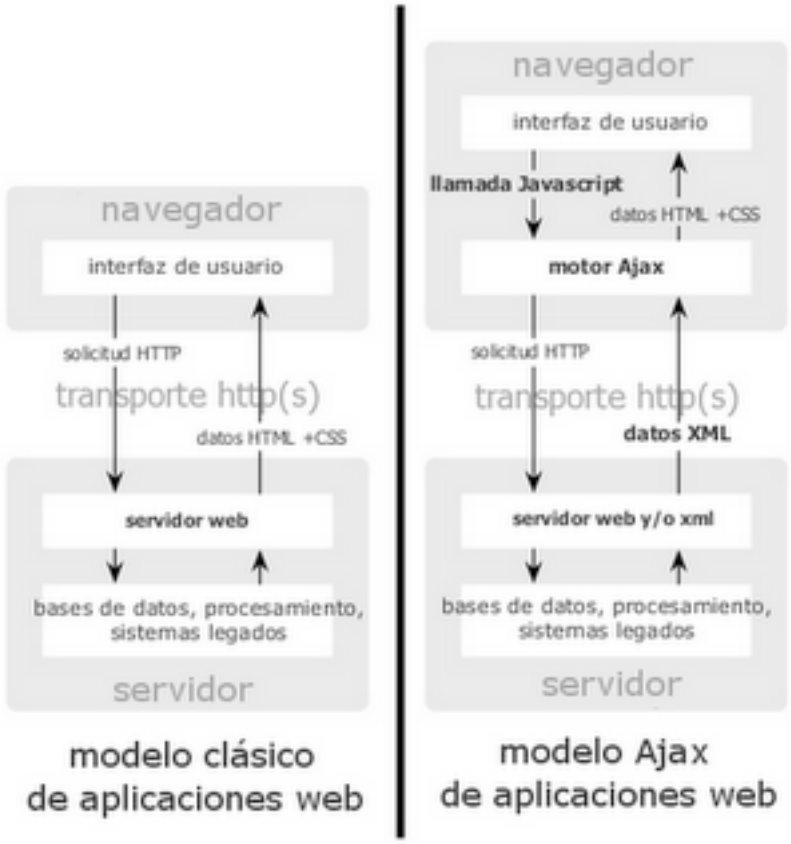


Figura 24: Modelo de aplicaciones web [23]

Las peticiones http al servidor se sustituyen por peticiones JavaScript que se realizan al Ajax-engine. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si se requiere una respuesta del servidor la petición se realiza de forma asíncrona a través de Ajax y el usuario no tiene porqué esperar la recarga de la página. En la siguiente figura se muestra una comparación entre los dos tipos de comunicaciones.

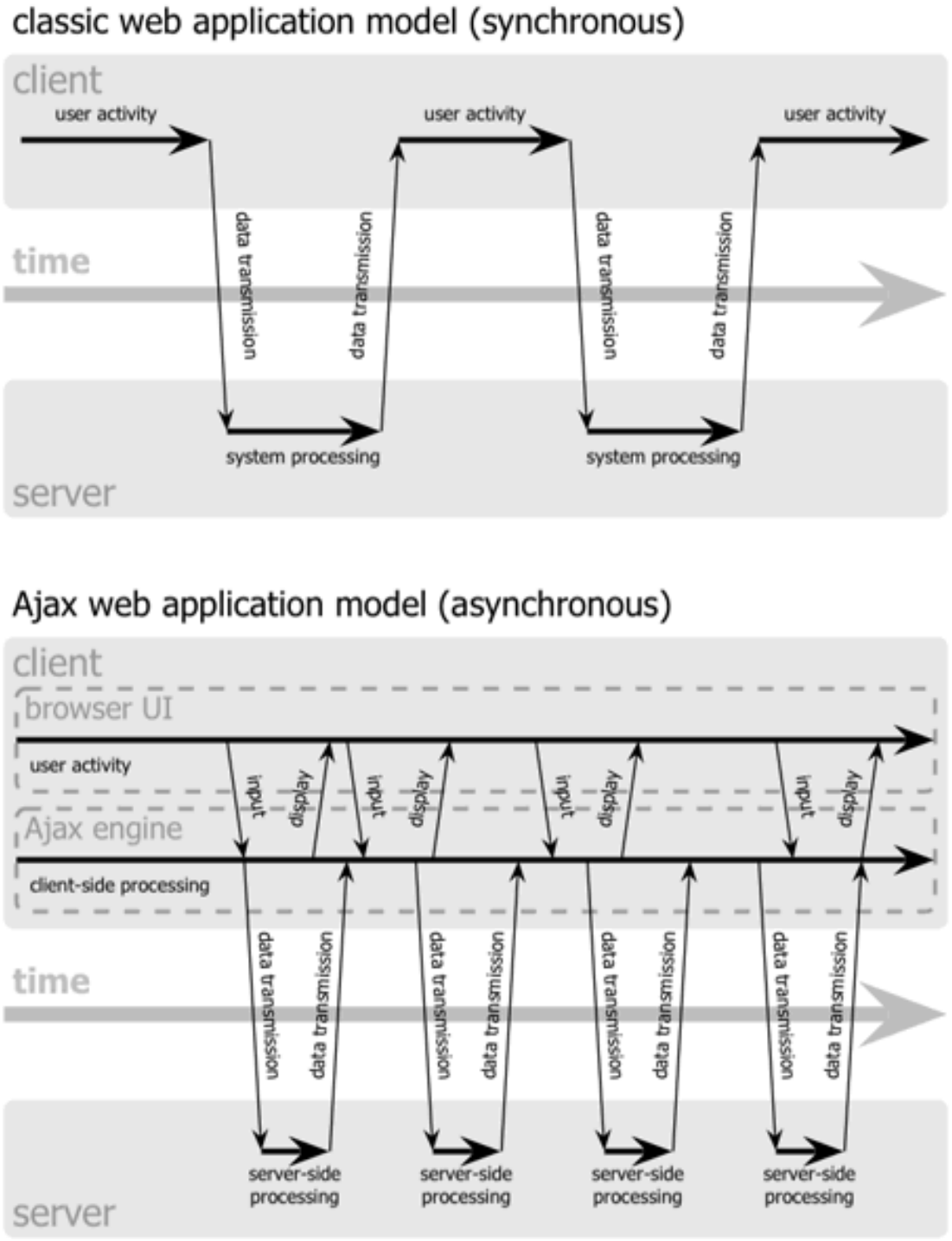


Figura 25: Peticiones síncronas y asíncronas [23]

Capítulo 3

Identificación de los requisitos del sistema

3.1 Introducción

En este capítulo del proyecto se detalla la estructura del sistema que se desea implementar. Se describen cada uno de los elementos que lo componen, así como las características principales y requisitos que deben cumplir dichos elementos.

Como se ha comentado anteriormente, el objetivo de este Proyecto Fin de Carrera es la implementación de un sistema de videollamada utilizando la tecnología WebRTC. Como se comentó en el apartado 2.2, WebRTC ofrece diversas posibilidades de implementación en el plano de señalización . A este respecto, para el desarrollo de este sistema se ha elegido SIP como protocolo de señalización, el cual, como se comentó en el apartado 2.4 posibillita el establecimiento, modificación y liberación de sesiones multimedia. Adicionalmente, se impone como requisito de diseño adicional, el correcto funcionamiento del sistema sobre entornos de red IMS (véase apartado 2.3 de esta memoria), de modo que éste deberá operar de manera compatible con el perfil de SIP definido para IMS por el 3GPP.

3.2 Arquitectura del sistema

En la siguiente figura se puede ver la arquitectura final del sistema de videollamada desarollado.

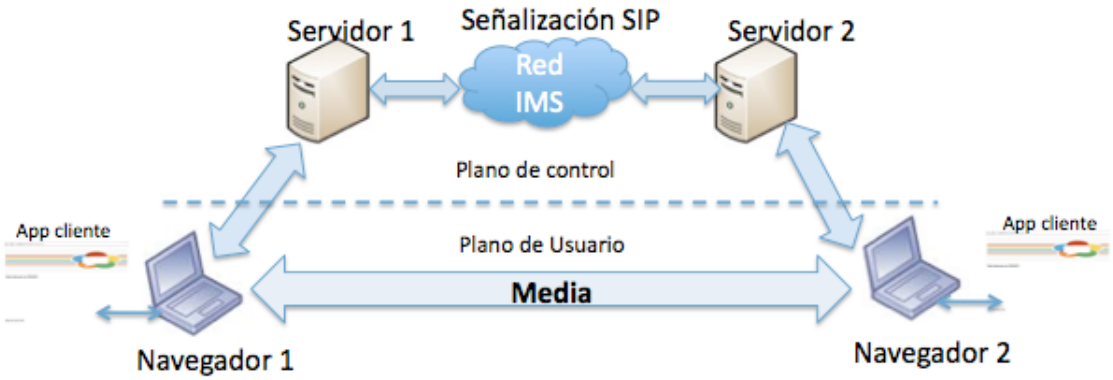


Figura 26: Arquitectura del sistema

Como se aprecia en la figura, el sistema diseñado consta de dos planos diferenciados, en línea con la estructura definida por el 3GPP para la arquitectura de la red IMS: un plano de control, que posibilitará el intercambio de información de control con el objetivo de establecer sesiones multimedia entre los puntos finales de la comunicación; y un plano de usuario, que posibilitará el intercambio bidireccional de la información multimedia (audio y vídeo) correspondiente a la videollamada.

El sistema consistirá en una aplicación cliente, que se ejecutará en el navegador Web del usuario utilizando el soporte ofrecido por WebRTC, que estará habilitado en dicho navegador. El usuario accederá al servicio de videollamada a través de un servidor Web, al que accederá mediante una URL bien conocida que introducirá en su navegador. Dicho servidor intercambiará información de control con la aplicación del cliente, y permitirá establecer (iniciar y recibir) videollamadas con otros usuarios del sistema, los cuales podrán acceder al mismo desde servidores Web diferentes. Para ello, cada servidor web incluye un Agente de Usuario compatible con el perfil de SIP definido por el 3GPP para redes IMS [37], lo que permite establecer las sesiones multimedia a través de las entidades CSCF de dicho subsistema. El intercambio de audio y vídeo correspondiente a una videollamada establecida se realizará directamente entre las aplicaciones cliente de los usuarios involucrados en la llamada, no siendo necesaria la utilización de las entidades funcionales de IMS a tal efecto.

A continuación se detallan los requisitos y funcionalidades que debe satisfacer cada uno de los componentes del sistema anteriormente descritos.

3.3 Aplicación de Cliente

Como se ha dicho la aplicación de cliente se ejecutará directamente en el navegador, permitiendo al usuario interactuar con el servicio implementado. A continuación se detallan los requisitos y funcionalidades que debe cumplir la interfaz de usuario, el plano de control y el plano de usuario.

3.3.1 Interfaz del usuario

La interfaz de usuario que se carga en el navegador permite ejecutar las siguientes acciones:

- Cuando el cliente web se carga por primera vez en el navegador deberá registrarse correctamente en la red IMS, para ello cuando se presione el botón "Login" se le solicitará al usuario un nombre y una contraseña para poder registrarlo en la red IMS. Este proceso de registro permite autenticar al usuario en la red IMS y que el nodo S-CSCF del Core IMS conozca la ubicación del cliente y la tenga almacenada para las llamadas que se ejecuten.

- Una vez realizado el registro del usuario, el cliente web ofrece la posibilidad al usuario de iniciar una videollamada o bien de quedarse a la espera de recibirla.

3.3.2 Funcionalidad del plano de control

El plano de control de la aplicación del cliente se centrará en la gestión y ejecución de las funciones necesarias para el establecimiento de sesión entre usuarios.. Este plano deberá cumplir los siguientes requisitos:

- El usuario deberá poder registrarse en el CORE IMS indicando un nombre y una contraseña correctas. Para ello será capaz de establecer comunicación con el servidor web de la aplicación, el cual iniciará el registro con la información recibida del cliente.

- El cliente web será el encargado de gestionar las descripciones locales y remotas para el correcto funcionamiento de WebRTC. Se encargará de generar las correspondientes ofertas y respuestas, con las cargas SDPs necesarias para el óptimo funcionamiento de WebRTC. Para ello se apoyará en el protocolo ICE para la generación de los candidatos de conexión que irán integrados en las distintas cargas SDP.

- Si el usuario decide iniciar una llamada deberá poder indicar el usuario con el que quiere establecer la comunicación. Esta información será enviada al servidor web de la aplicación el cual se encargará de la gestión de la señalización. El usuario con el que se desea contactar debe estar

correctamente registrado en el HSS del Core IMS ya que si no la comunicación no se establecerá.

- El plano de control será capaz de gestionar las llamadas entrantes, manteniendo una comunicación constante con el servidor web.

3.3.3 Funcionalidad del plano de usuario

El plano de usuario debe cumplir los siguientes requisitos para validar la implementación de la aplicación que se pretende desarrollar en este Proyecto Fin de Carrera:

- El cliente web será el encargado de gestionar la comunicación multimedia, ya que como se ha explicado anteriormente la comunicación establecida es directa entre los navegadores.

- El plano de usuario se encargará del envío y recepción del audio y vídeo de la llamada establecida.

3.4 Servidor Web

3.4.1 Funcionalidad del plano de control

Los servidores Web que se pueden ver en la figura anterior actuarán como Agentes de Usuario SIP encargados del control de los mensajes de señalización para el registro de los usuarios en la red IMS y para el establecimiento de sesiones entre los dos extremos de la comunicación.

El protocolo de señalización empleado es SIP (véase sección 2.4 de este documento). Para el desarrollo de los mensajes SIP se ha empleado el *API JAIN-SIP* (véase sección 2.12 de este documento), el cual es un conjunto de librerías de Java que permite el uso y fácil manejo del protocolo SIP.

Las funcionalidades de los servidores web son:

- Deben realizar el registro del usuario al iniciarse el cliente web. Deben intercambiar los mensajes de la siguiente figura con la red IMS para el correcto registro del usuario.

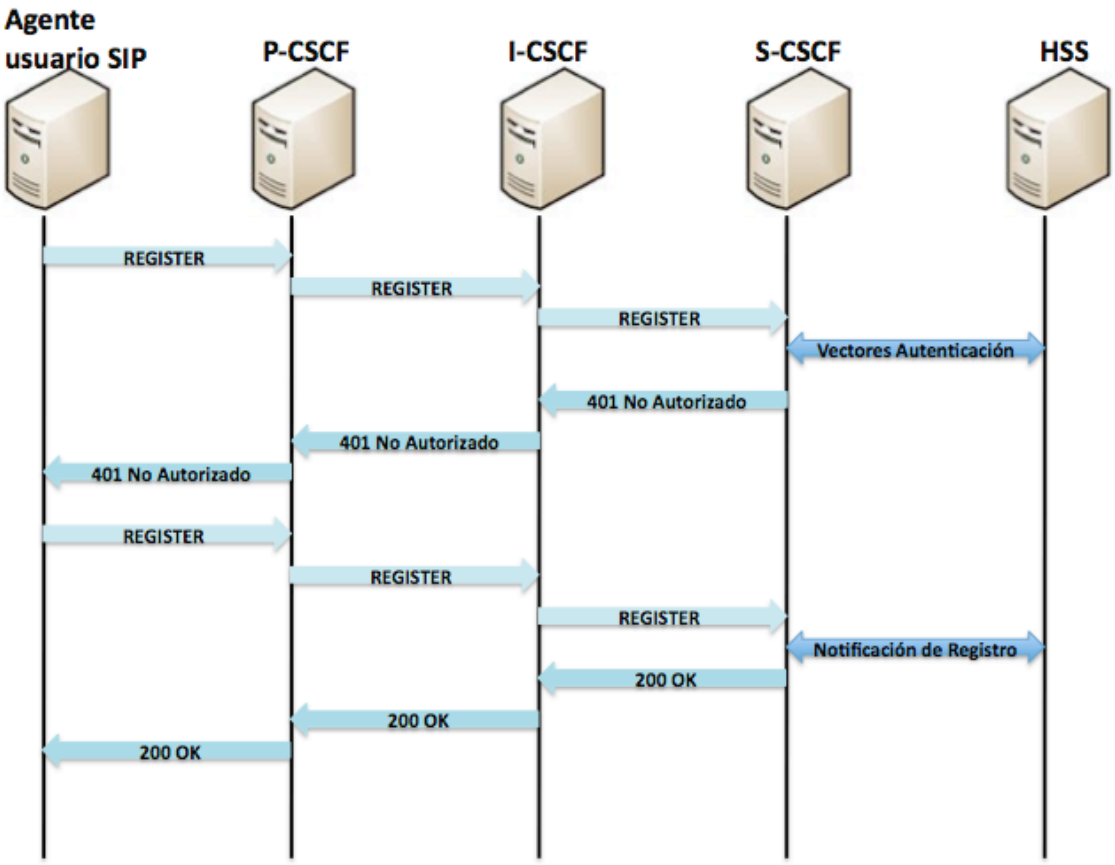


Figura 27: Registro en la red IMS

Como se puede ver en la figura el registro se hace en cuatro pasos diferenciados. Primero se genera una petición de registro la cual entra a la red IMS por el nodo P-CSCF, el cual es el nodo de contacto entre la red IMS y el terminal, como se comentó en el apartado 2.3.3.2. Este primer mensaje Register pasa por la red IMS como se muestra en la figura hasta llegar al S-CSCF, el cual intercambiará los datos de autenticación con el HSS por medio del protocolo DIAMETER. La primera respuesta del HSS es un No autorizado. Esta respuesta que se envía de vuelta al nodo que ha iniciado el registro lleva un reto en su interior, el cual deberá ser resuelto por el nodo que inició el registro y devuelto al HSS para su comprobación. Cuando el HSS recibe por segunda vez los datos para la autenticación que llegan al S-CSCF comprueba el contenido de los mismos asegurándose de que el reto propuesto ha sido resuelto de forma satisfactoria. Si esto es así se devolverá, el usuario quedará registrado en la red IMS recibiendo por parte de los nodos CSCF una respuesta del tipo 200 OK .

- Establecimiento de sesión entre el navegador que inicia la conexión y el navegador que la recibe. Cuando un cliente web decide iniciar la comunicación el agente usuario SIP debe gestionar los mensajes de señalización de forma correcta con la información que le llega de la aplicación de cliente. En la siguiente figura se puede ver el intercambio de mensajes necesarios para el establecimiento de sesión entre dos navegadores.

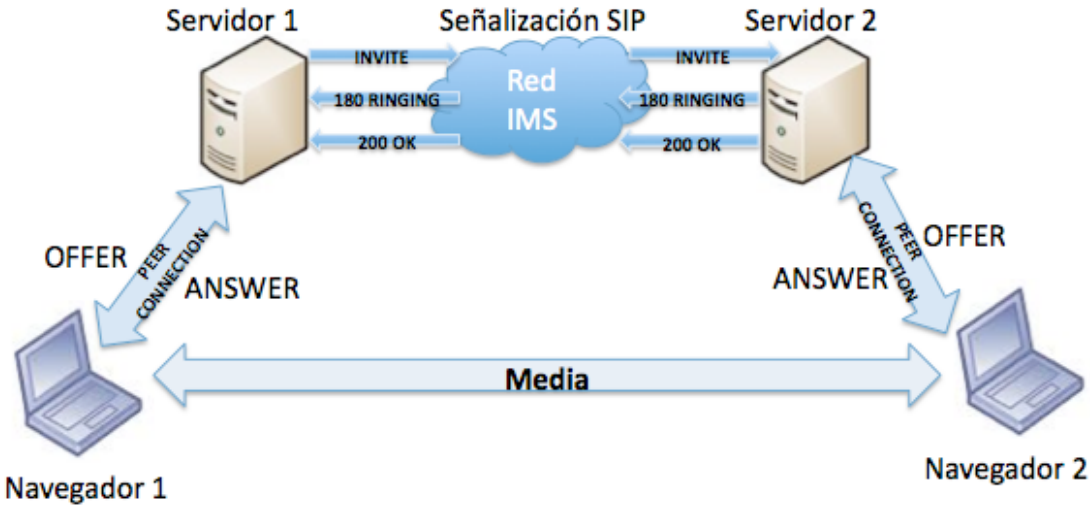


Figura 28: Establecimiento de sesión entre navegadores

Cuando la aplicación de cliente que está corriendo en el navegador 1, una vez el usuario ha sido registrado en la red, decide iniciar la comunicación genera una oferta SDP que envía al Servidor 1. Este servidor 1, haciendo el papel de agente usuario SIP, generará la petición de INVITE que será lanzada a la red IMS. El servidor 2, haciendo el mismo papel que el servidor 1, recibirá esta oferta y pasará los datos oportunos de la petición recibida a su navegador. El navegador 2 en el que estará corriendo la aplicación de cliente establecerá la oferta SDP recibida como descripción remota y devolverá a su servidor su descripción local para que sea enviada en un mensaje 200 OK para establecer la comunicación. Este mensaje 200 OK pasará por la red IMS hasta llegar al servidor 1. Este servidor pasará los datos recibidos a su navegador que establecerá el SDP recibido como descripción remota. Entonces el Agente usuario SIP generará un mensaje de tipo ACK para validar la conexión. Una vez que ambos lados conocen las descripciones SDP del lado contrario queda establecida la comunicación directa entre navegadores.

Capítulo 4

Implementación

4.1 Introducción

La aplicación de videollamada que ha sido desarrollada en este Proyecto Fin de Carrera se compone de dos módulos diferenciados: La aplicación del cliente, que se ejecuta en el navegador web del usuario, y el servidor web que también ejerce el papel de agente usuario SIP/IMS. La aplicación del cliente será la encargada de gestionar las peticiones del usuario, así como de controlar la captura, envío, recepción y reproducción de los flujos de audio y vídeo una vez la videollamada esté establecida. Esta aplicación del cliente será acessible única y exclusivamente mediante navegadores Google Chrome en una versión igual o superior a la 29. Esta aplicación interactúa con un servidor web que ejercerá de Agente Usuario SIP.

El sistema completo que ha sido implementado se muestra en la figura 26 y se puede estructurar en dos planos diferenciados:

- Plano de control: Por una parte este plano se encarga de la gestión de la señalización para el establecimiento de sesión entre los dos extremos de la comunicación. El protocolo de señalización elegido para el desarrollo de este Proyecto Fin de Carrera es SIP. Dicho protocolo permite el establecimiento de sesiones multimedia a través de la red IMS. Los mensajes SIP son intercambiados entre los dos servidores web que hacen el papel de Agente usuario SIP, a través del Core IMS . Por otro lado el plano de control se encargará de la gestión de los recursos de WebRTC en la aplicación de cliente.

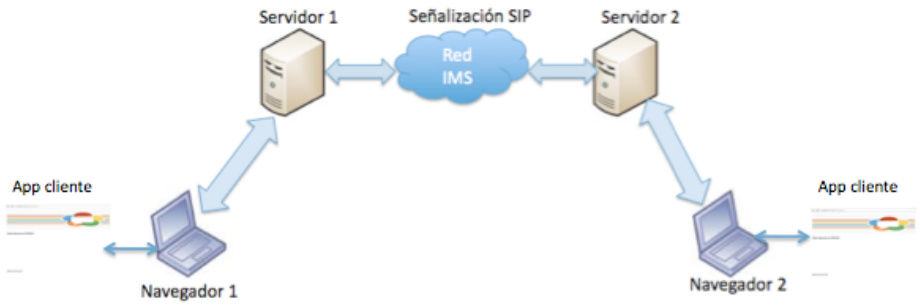


Figura 29: Plano de Control

- Plano de usuario: basado en la comunicación directa entre las aplicaciones de cliente corriendo en los dos navegadores. Una vez queda establecida la sesión entre los dos navegadores los datos de audio y video de la llamada se intercambian directamente entre ellos sin involucrar al plano de control y por lo tanto sin involucrar a los Agente Usuario SIP. Esto lo vemos en la siguiente figura.

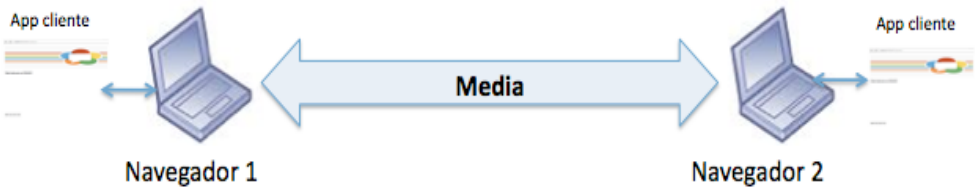


Figura 30: Plano de Usuario

A continuación, se describen cada uno de los elementos involucrados en el desarrollo, así como las tecnologías utilizadas para su implementación.

4.2 La aplicación del Cliente

La aplicación cliente está formada por una interfaz gráfica y un módulo de procesamiento de videollamadas. La interfaz gráfica se ha desarrollado utilizando HTML y HTML 5 (ver apartado 2.9.3). Para el módulo de procesamiento de llamadas se ha desarrollado una clase Javascript (ver apartado 2.9.2) que emplea la tecnologíaa AJAX (ver apartado 2.13) para el envío de datos al servidor web.

En la siguiente tabla se hace un breve resumen de las clases implementadas en el cliente web, asi como un resumen de su funcionalidad principal.

Nombre de la Clase	Tipo	Descripción
rtcweb	HTML/HTML5	Define la interfaz gráfica que hace de intermediaria entre el usuario y la aplicación.
main	Javascript	Define el módulo de procesamiento de video llamadas.

Tabla 6: Conjunto de clases que forman el cliente web

4.2.1 La interfaz gráfica

Como se ha dicho la interfaz gráfica está implementada el HTML/HTML 5. Cuando el usuario carga la aplicación por primera vez lo visualiza es la siguiente figura.

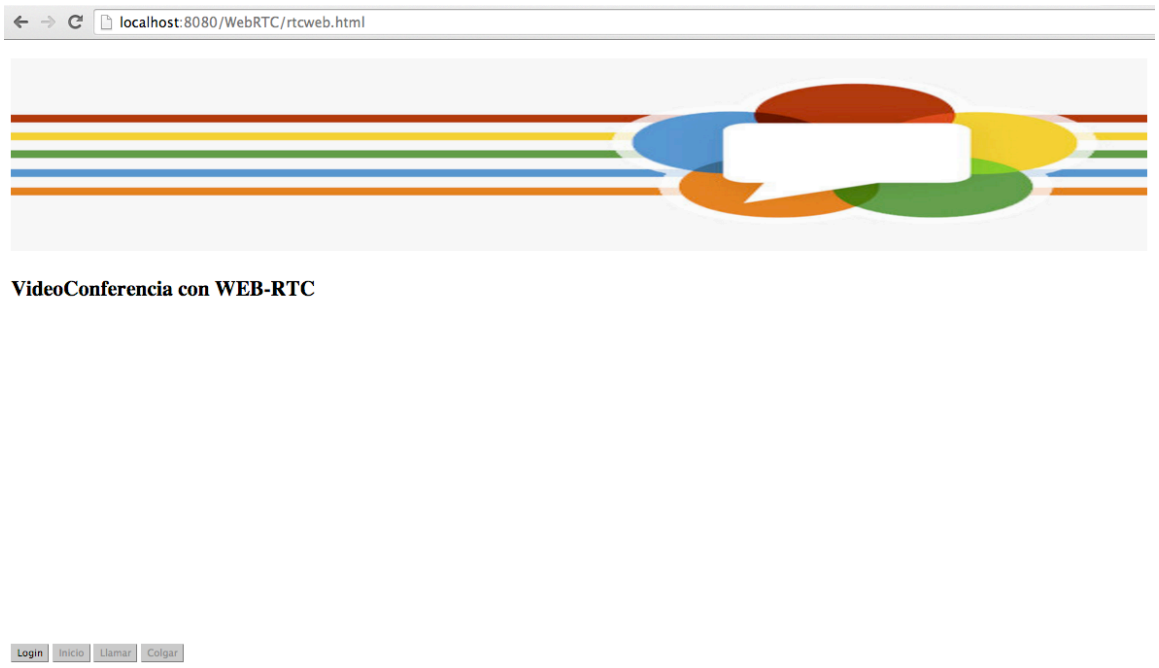


Figura 31: Interfaz gráfica de la aplicación

El único botón habilitado al cargar la aplicación por primera vez es el botón de Login Este botón de Login de la aplicación permite iniciar el registro del usuario en la red IMS, para ello solicitará al usuario un nombre y una contraseña que enviará al servidor web el cual iniciará el proceso de registro en la red IMS. En las siguientes imágenes podemos ver los datos que solicita la aplicación para el registro que se explicará en detalle en la sección 4.3.1.

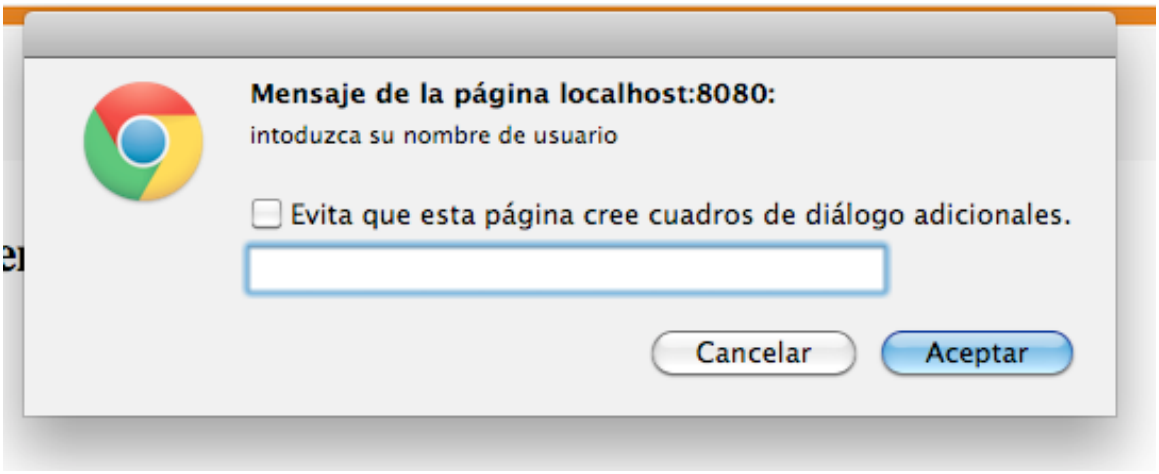


Figura 32: Petición de nombre de usuario

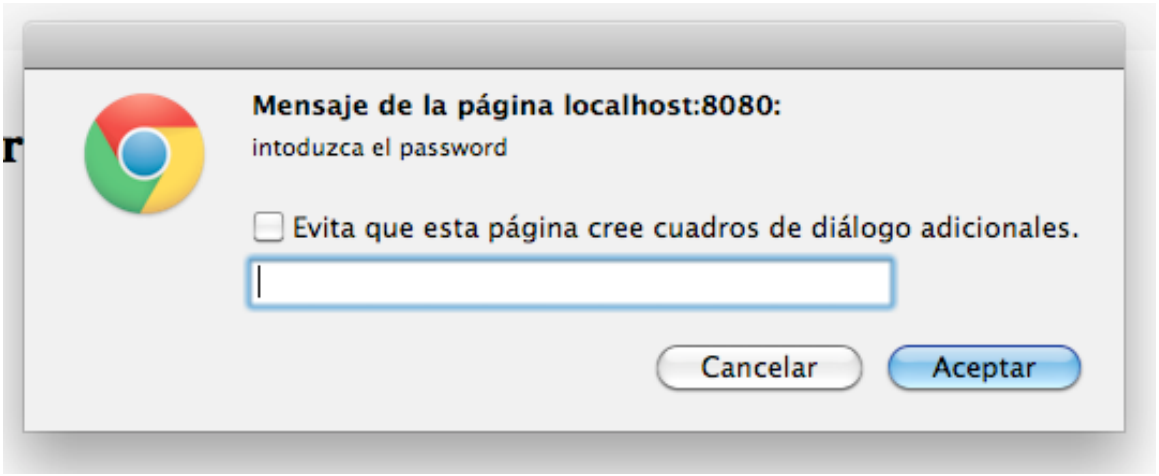


Figura 33: Petición de contraseña

Después de registrarse de forma correcta en la red IMS el usuario ya estará listo para realizar o recibir una videollamada. El único botón que tendra habilitado en la interfaz será el botón de inicio, el cual al ser presionado solicitará al usuario su consentimiento expreso para acceder al audio y vídeo local.

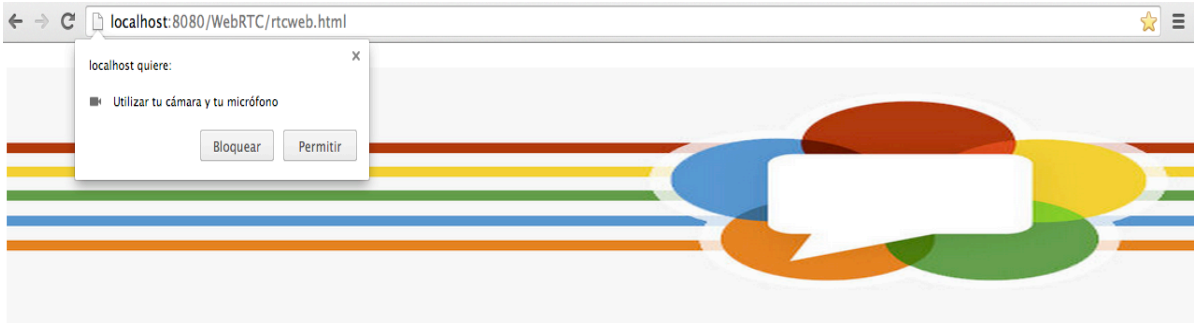


Figura 34: Petición de autorización de acceso a audio y vídeo local

Una vez el usuario a concedio permiso a la aplicación para que tenga acceso al audio y vídeo local, el usuario podrá iniciar una vídeo llamada. Si el usuario decide iniciar una videollamada, la aplicación le solicitará que introduzca el nombre del usuario con el que desea establecer la sesión.

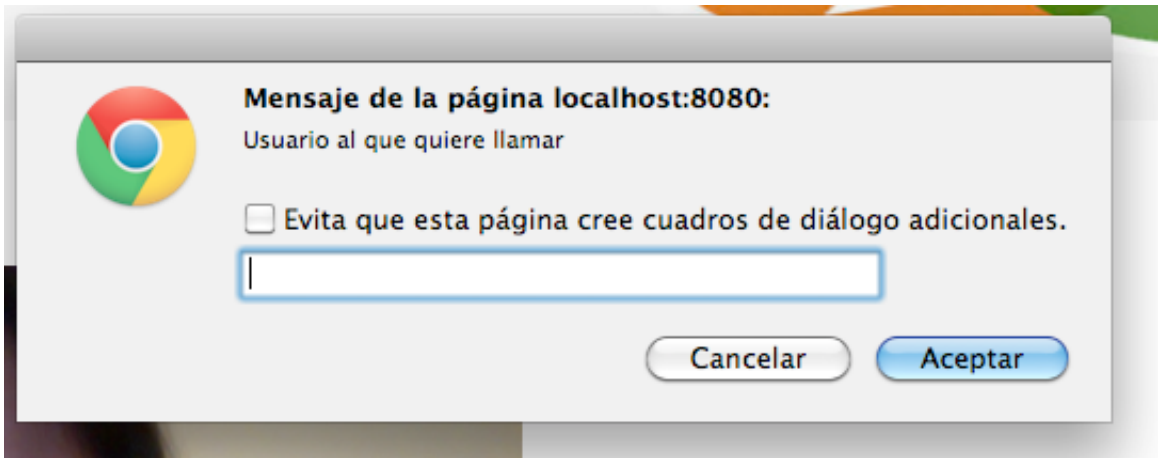


Figura 35: Petición de usuario llamado

Lo mismo ocurrirá para el intercambio de descripciones SDP entre navegador y servidor web. Toda esta información será intercambiada a través del servlet.

4.2.2 Módulo de procesamiento de videollamadas

Por último, la aplicación cliente web incluye un módulo de gestión de llamadas basado en Javascipt y en la tecnología WebRTC. El Javascript en su fichero main.js implementa funcinalidades del plano de control y del plano de usuario. Aquí se implementará toda la lógica para la generación y gestión de las ofertas y respuestas SDP necesarias para cumplir las especificaciones de WebRTC.

Será el encargado de manejar eventos y mantendrá una comunicación abierta de forma permanente con el agente usuario SIP, el cual estará ubicado en el servidor web. La gestión de las ofertas y respuestas se realizará de forma distinta si el usuario es el que comienza la llamada o es el que la recibe.

Figura 36: Gestión de descripciones en Web-RTC

A continuación se detalla todo el proceso seguido por WebRTC para el establecimiento de sesiones entre usuarios. Como se acaba de comentar la gestión será distinta si el usuario es el que inicia la llamada o es el que la recibe, por tanto se hará un análisis de cada uno de los casos.

4.2.2.1 Usuario llamante

Cuando el usuario decide iniciar la llamada el proceso seguido para el correcto funcionamiento de WebRTC es el que queda reflejado en el siguiente diagrama.

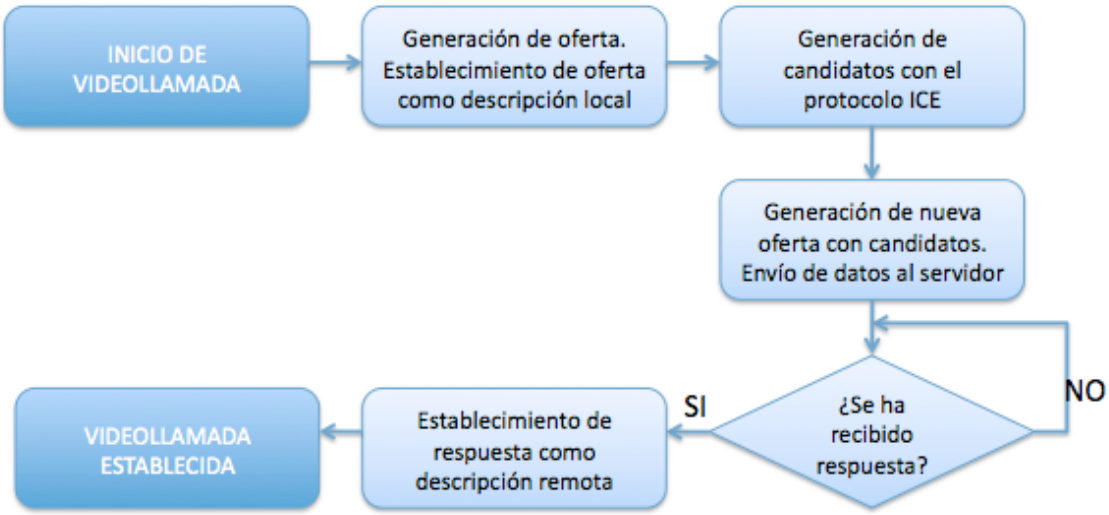


Figura 37: Diagrama de flujo usuario llamante

En el momento en el que se decide iniciar la llamada el Javascript genera una oferta SDP la cual se establece como descripción local del usuario llamante. Así mismo en el momento en el que se establece la oferta local se hace uso del protoclo ICE (véase sección 2.7 de este documento) para obtener una lista de candidatos de conexión. Al recibir todos estos candidatos se creará una nueva oferta SDP la cual será pasada al servidor web, por medio de AJAX. El servido web, como se verá en la siguiente sección, se encargará de gestionar la señalización con el otro extremo. Cuando recibe las respuestas pertinentes las pasará de nuevo a la aplicación del cliente. El Javascript establecerá esta respuesta recibida como descripción remota y la comunicación quedará establecida.

4.2.2.2 Usuario llamado

Cuando el usuario es el que recibe la llamada el proceso seguido para el correcto funcionamiento de WebRTC es el que queda reflejado en el siguiente diagrama.

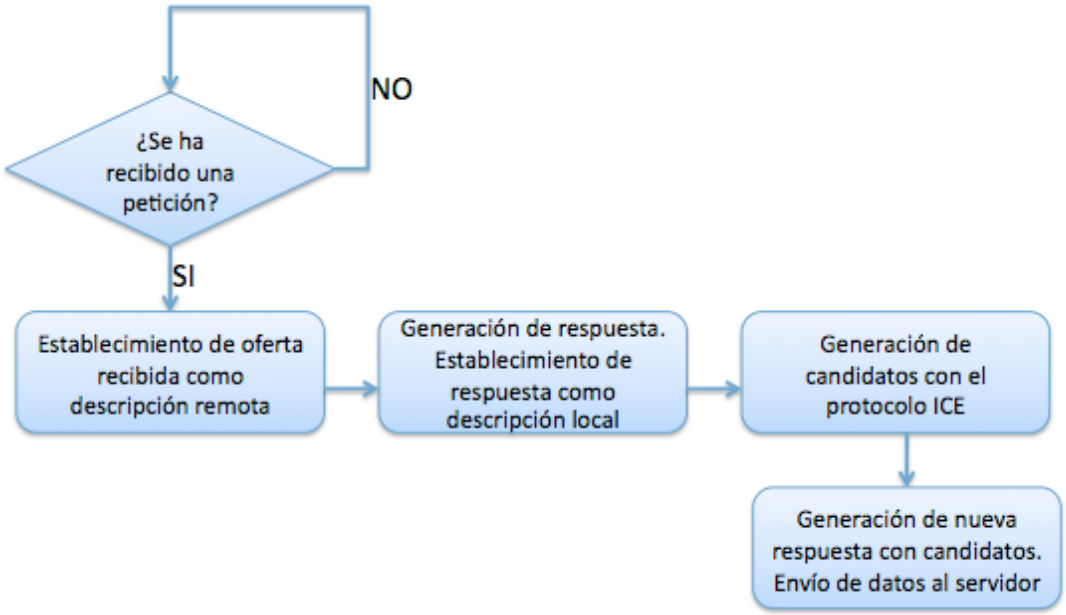


Figura 38: Diagrama de flujo usuario llamado

Como se ha comentado en esta misma sección el Javascript tiene una conexión abierta de forma constante con el servidor web, por tanto cuando el servidor web recibe la petición de establecer la comunicación le pasará los datos recibidos en la petición al Javascript mediante el servlet de la aplicación. Cuando el Javascript reciba los datos del inicio de la conexión establecerá la oferta SDP recibida como descripción remota y generará una respuesta la cual establecerá como descripción local. De la misma manera que para el extrema que inicia la videollamada en el momento en el que se establece la descripción local se hace uso del protoclo ICE (véase sección 2.7 de este documento) para obtener una lista de candidatos de conexión. Una vez obtenidos todos los candidatos se generá una nueva respuesta que contenga la información que ha sido proporcionada por el protocolo ICE. Esta respuesta se enviará al servidor web por medio de AJAX, para que dicho servidor se encargue de pasar la información al otro extremo de la comunicación como se verá en los siguientes apartados de este documento.

4.3 El Servidor Web

El servidor web se encarga de gestionar las peticiones que le llegan de la aplicación de cliente por parte del usuario a través de la interfaz gráfica mediante el Servlet. Además el servidor web ejercerá el papel de Agente usuario SIP. Este agente usuario SIP está formado por dos clases Java que se encargarán de la gestión de los mensajes SIP con el fin de hacer la petición de registro del usuario en la red IMS y gestionar la señalización de la videollamada. En la siguiente tabla se hace un breve resumen de las clases que integra este servidor web.

Nombre de la Clase	Tipo	Descripción
Rtc	Java	Servlet de la aplicación. Recibe las peticiones del cliente web.
web	XML	Descriptor XML del Servlet Le indica al servidor Apache tomcat la ubicación del Servlet.
Register	Java	Realiza el registro del usuario en la red IMS
SipLayer	Java	Se encarga de la generación de los mensajes SIP para el establecimiento de sesión entre usuarios.

Tabla 7: Conjunto de clases que forman el Agente Usuario SIP

Para implementar la señalización SIP, se ha hecho uso de API de Java *JAIN SIP* (ver apartado 2.12). Gracias a esta API podemos gestionar los mensajes SIP de una manera fácil sin necesidad de analizar la compleja sintáxis de dichos mensajes. Para ello *JAIN SIP* utiliza un *SIP Provider*, que recibe las peticiones, las procesa transformándolas en eventos y entrega dichos eventos a la aplicación, y un *SIP Listener* que gestiona esos eventos recibidos para que la aplicación sea capaz de interpretarlos.

Las librerías que se han empleado para que el agente usuario SIP sea capaz de manejar lo descrito anteriormente se detallan a continuación. Cada una de estas librería ha sido cargada en el servidor *Apache Tomcat* (Véase sección 2.11 de este documento) para la correcta gestión de la señalización.

- **commons-codec-20041127.091804.jar**: Se utiliza para el registro del usuario en la red IMS.

- **jainSipApi1.2.jar** y **jainSipRi1.2.jar**: Implementa clases principales, interfaces de SIP y referencias de implementación.

- **concurrent.jar**: Se encarga de la gestión de procesos concurrentes de la aplicación.

- **log4j-1.2.8.jar:** Se encarga de la gestión de los logs de debug y los avisos de la aplicación.

A continuación se describen tanto el proceso de registro en la red IMS como el establecmiento de sesión entre usuarios, así cómo la funcionalidad del Servlet de la aplicación, encargado de enviar y recibir las peticiones necesarias para el registro y establecimiento de la conexión.

4.3.1 El Servlet

El Servlet desarrollado para la aplicación es del tipo HTTP. Dicho Servlet recibe las peticiones que llegan desde la aplicación de cliente al servidor web, las procesa y emite las respuestas correspondientes. Las peticiones y respuestas intercambiadas entre el cliente Web y el Servlet son del tipo *HTTPServletRequest* y *HTTPServletResponse*, respectivamente. Para poder hacer uso de este tipo de intercambio de datos será necesario tener integrado en el sistema la librería *servlet-api.jar*.

La aplicación de cliente hace uso del Servlet de la aplicación para el intercambio de datos entre el navegador en que se está ejecutando la aplicación y el servidor web. Un ejemplo se puede ver en el siguiente diagrama de flujo. Cuando un usuario carga por primera vez la página web y quiere registrarse en la red IMS, deberá enviar un nombre y una contraseña al servidor web para que éste pueda iniciar el registro. Esta traspaso de información se hará a través del servlet de la aplicación.

4.3.2 El Registro

La primera funcionalidad del agente usuario SIP es el registro del usuario en la red IMS una vez recibida la petición por parte del Servlet. El proceso que debe serguirse para la autenticación y registro del usuario se muestra en la figura 27. La implementación del registro del usuario en la red IMS no forma parte del desarrollo de este proyecto, se hace uso de una librería que se desarrolló en el siguiente documento "Desarrollo de un servicio P2P TV para redes de Internet de próxima generaciónn" [34], pero es imprescindible su explicación y comprensión para entender el funcionamiento de la aplicación final. El proceso que sigue un usuario para registrarse en la red se muestra en el siguiente diagrama.

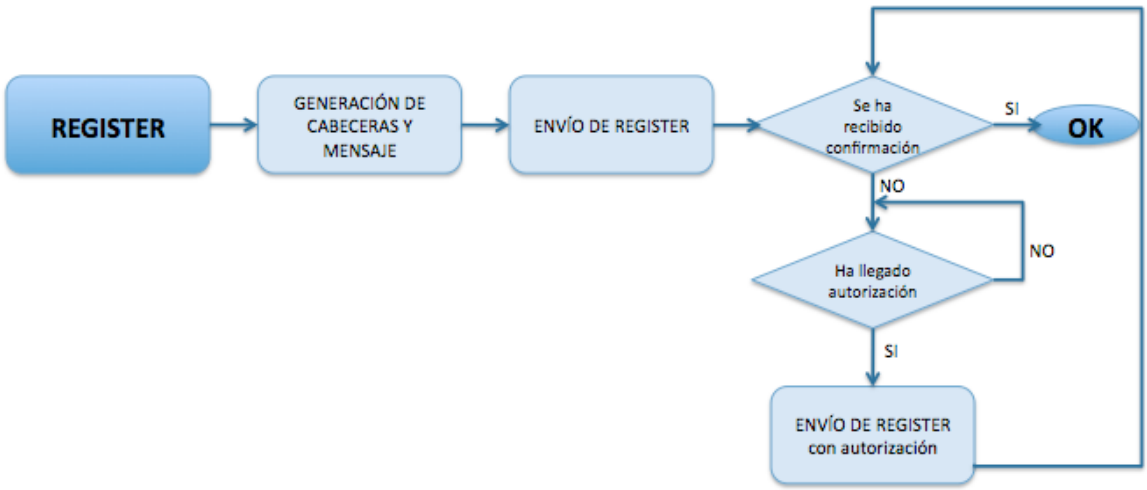


Figura 39: Diagrama de registro en la red IMS

Como se comentó con anterioridad, en la sección 3.4.1 de este documento, el registro del usuario se hace en 4 pasos, una vez que el agente usuario SIP recibe la petición de registro por parte del Servlet se generan las cabeceras y el mensaje Register. En el

mensaje Register generado por el agente usuario SIP se creará una cabera Route que obligará al mensaje a encaminarse a la red IMS, donde el primer punto de contacto será el P-CSCF. Una vez el mensaje llegue a la red IMS pasará por los nodos señalados en la figura 27. La primera vez que el HSS reciba esta petición para autenticar al usuario su respuesta será un No autorizado que contendrá un reto que debe ser resuelto por el usuario para autenticarse en la red. El P-CSCF enviará esta respuesta al agente usuario SIP que enviará un nuevo registro con el reto resuelto para poder autenticarse. El HSS validará este mensaje y enviará de vuelta un respuesta de 200 OK validando el registro del usuario.

4.3.3 Establecimiento de sesión

Una vez concluído con éxito el registro del usuario en la red IMS por parte del agente usuario SIP, el usuario puede iniciar una vídeo llamada o esperar a recibirla. Si el usuario decide iniciar una videollamada debe enviar al Agente Usuario SIP el nombre del usuario con el que desea establecer la comunicación, así como su descripción SDP para que el Ageste Usuario SIP pueda generar la petición de INVITE e iniciar el proceso de señalización. Esto queda reflejado en el siguiente diagrama de flujo.

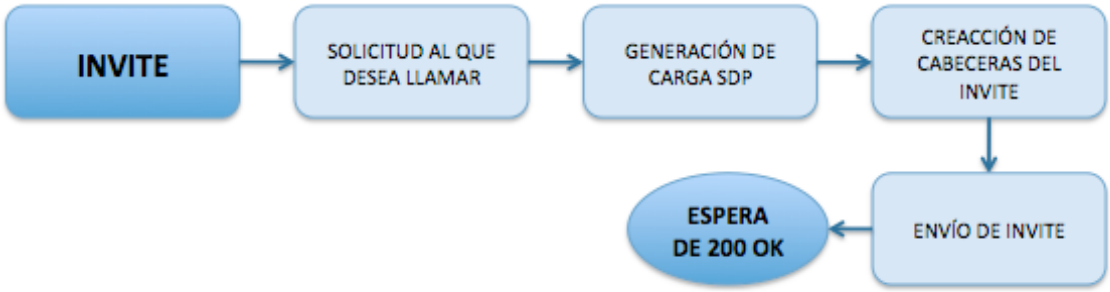


Figura 40: Diagrama de flujo generación INVITE

Cuando el usuario recibe la respuesta esperada, 200 OK, por parte del otro extremo de la comunicación, debe procesarla, pasando la información conveniente a su aplicación de cliente, y confirmarla generando un solicitud de ACK. Esto se muestra en el siguiente diagrama de flujo.

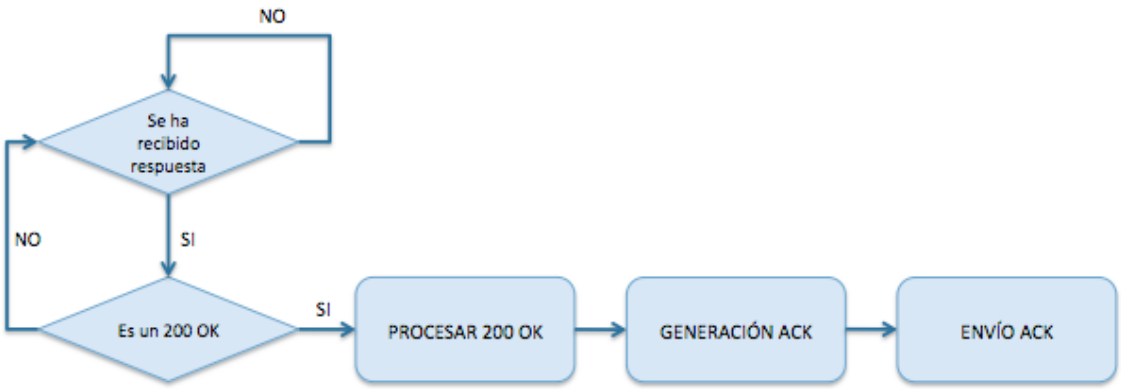


Figura 41: Procesamiento de 200 OK

Por otro lado está el usuario que recibe la llamada. El Agente usuario SIP de este usuario será el que reciba la petición de INVITE que ha sido enviada por el Agente Usuario SIP del llamante. Para recibir y procesar la Petición de INVITE el agente usuario SIP se mantiene a la espera de recibir una petición.

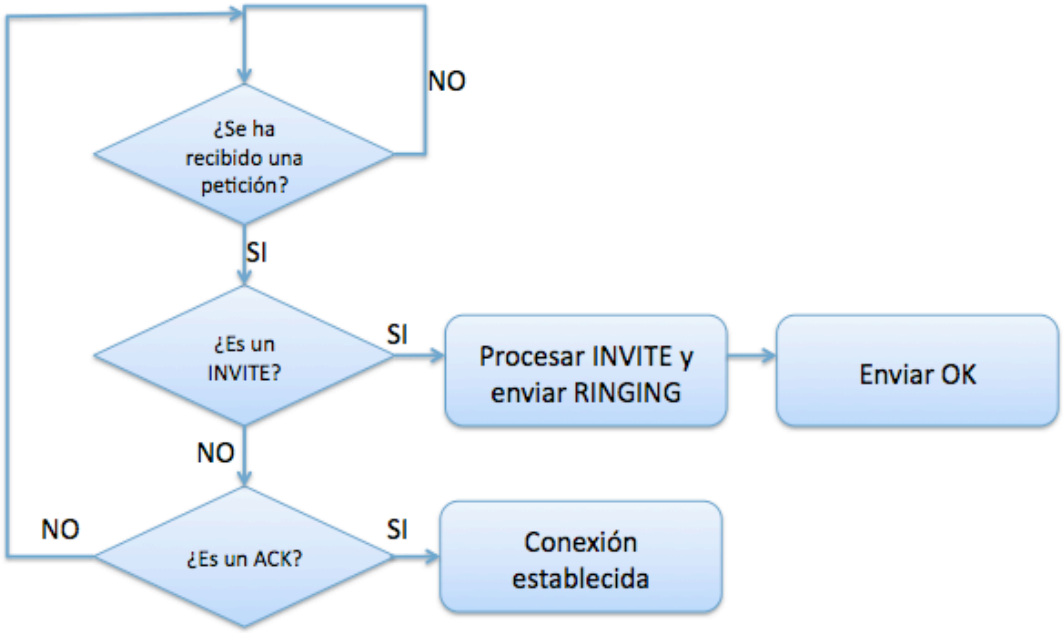


Figura 42: Procesamiento de peticiones

Capítulo 5

Pruebas

5.1 Introducción

En este capítulo se describen con detalle las pruebas realizadas sobre el sistema implementado para poder verificar su correcto funcionamiento. Como su propio título indica, el objetivo de este Proyecto Fin de Carrera es el desarrollo de una aplicación de videollamada basada en SIP y Web-RTC. Adiccionalmente en las pruebas se ha utilizado una implemtanción *opensource* de la arquitectura IMS [32] para la gestión de la señalización SIP. Este core IMS se encuentra disponible en los laboratorios del Departamento de Ingeniería Telemática en la universidad Carlos III de Madrid, y consta de los elementos necesarios para el control del estado de la sesión, es decir de los nodos CSCF (P-CSCF, I-CSCF y S-CSCF) y del nodo HSS para el registro de usuarios. En la siguiente figura podemos ver el escenario utilizado para las pruebas.

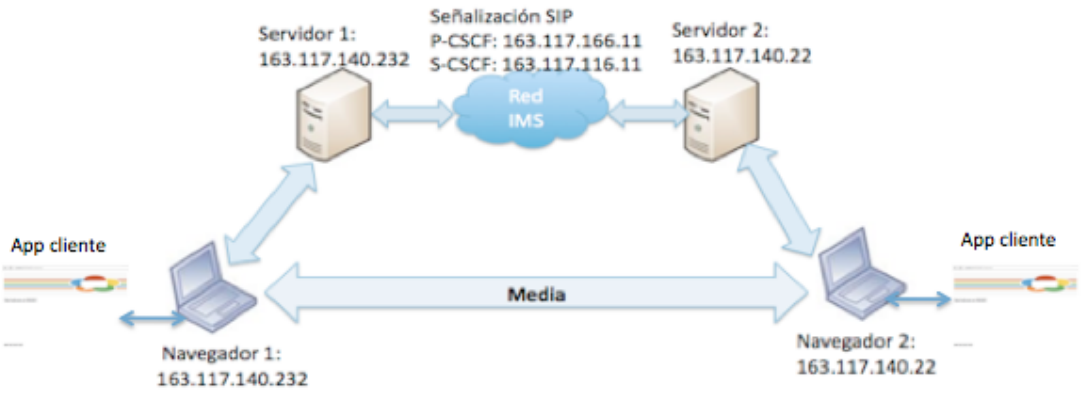


Figura 43: Escenario de Pruebas

En la siguiente tabla podemos ver los datos más relevantes de los elementos implicados en estas pruebas.

Elemento	Dirección IP	Puerto SIP
Cliente WEB-1 (p2pIPTV)	163.117.140.232	5060
Agente Usuario 1	163.117.140.232	--
Cliente WEB-2 (bob)	163.117.140.22	5060
Agente Usuario 2	163.117.140.22	--
P-CSCF	163.117.166.11	4060
S-CSCF	163.117.166.11	6060

Tabla 8: Especificación de los elementos que intervienen en las pruebas

Como podemos ver en la figura anterior podemos dividir el escenario de pruebas en tres partes:

- El primer conjunto formado por el navegador 1 y el primer servidor que hará de Agente usuario SIP. El navegador 1 será Google Chrome en su versión 47 y en él se ejecutará la aplicación del cliente y el primer servidor web será Apache tomcat en su versión 7.
- El segundo conjunto formado por el Core IMS en el cual se incluyen un conjunto de nodos CSCFs y el HSS como se ha detallado anteriormente.
- El tercer conjunto formado por el navegador 2 y el segundo servidor que hará de Agente usuario SIP. El navegador 2 será Google Chrome en su versión 47 y en él se ejecutará la aplicación del cliente y el segundo servidor web será Apache tomcat en su versión 7.

Para facilitar las pruebas, el navegador y el Agente usuario SIP se ejecutan desde el mismo equipo, aunque podrían ejecutarse en equipos diferentes.

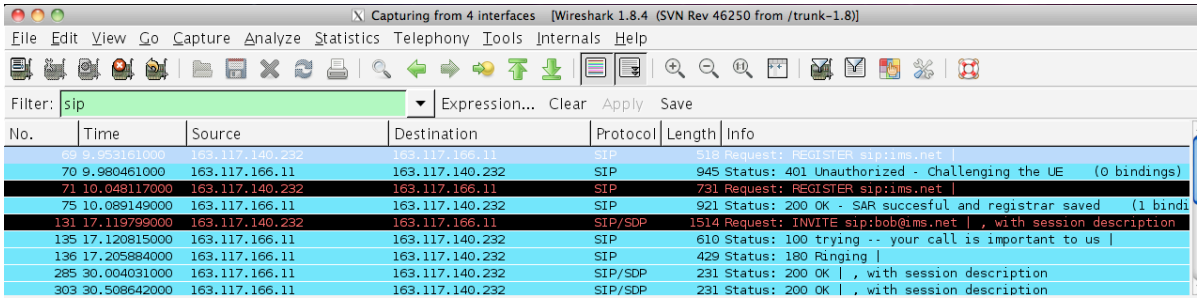
La prueba realizada para la validación de este Proyecto Fin de Carrera es el establecimiento de una videollamada entre los dos navegadores que se encuentran en el sistema. Se analizará esta videollamada en tres fases para ir validadando todo lo descrito anteriormente.

1. La primera fase constará del registro de usuarios en la red IMS. Se probará a registrar un usuario dado de alta en el HSS viendo los mensajes SIP intercambiados para dicho registro. Estos mensajes deben ser los indicados en el apartado 3.4.1 de esta memoria. Se probará a registrar un usuario que no esta dado de alta en el HSS y se comprobará que este registro es erróneo.

2. La segunda fase para el establecimiento de la videollamada será comprobar que la aplicación puede acceder al audio y vídeo local. Se debe comprobar también que la aplicación implementada solicita permiso expreso del usuario para poder acceder a estos dos flujos y poder transmitirlos posteriormente.

3. La tercerá y última fase para la validación de todo lo propuesto en este documento será el establecimiento de una sesión multimedia (audio y vídeo) entre los equipos de dos usuarios. Se probará el correcto establecimiento de las descripciones SDP locales y remotas en ambos navegadores, asi cómo el correcto intercambio de mensajes de señalización. La prueba concluirá con la transmisión y recepción de audio y vídeo por parte de los dos navegadores implicados.

El conjunto de mensajes intercambiados en las pruebas para la validación del correcto funcionamiento de la aplicación se pueden ver a continucación:



En los siguientes apartados de este documento se irán analizando dichos mensajes para comprobar el correcto funcionamiento de la aplicación con todo lo descrito en los apartados anteriores.

5.2 Registro de usuarios en la red IMS

Antes del establecimiento de sesión entre usuarios, cada usuario debe registrarse correctamente en la red IMS. La primera vez que se carga el cliente web la única posibilidad que se le ofrece al usuario es la de registrarse. Para ello el único botón que tiene habilitado en la interfaz gráfica de la aplicaciónn es el de “Login” y cuando lo presione se le pedirá al usuario que introduzca un nombre y una contraseña para la realización de dicho registro.

Para la verificación del correcto funcionamiento del registro se harán dos pruebas diferentes, se intentará registrar un usuario dado de alta correctamente en el HSS, este registro debe terminar de manera exitosa tal y como se vio en el apartado 3.4.1 y

seguidamente se intentará registrar un usuario que no está dado de alta en el HSS, por lo que el registro no se completará de manera exitosa.

La primera prueba que se realiza es el registro de un usuario correctamente dado de alta en el HSS. Como se indicó en el paratado 3.4.1 el registro se realiza en 4 fases, las cuales se analizan con las siguientes figuras.

Para ejecutar el registro de forma correcta cuando se ejecuta el botón "Login" de la aplicación, ésta solicitará un nombre de usuario y una contraseña con la que intentará ejecutar el registro en el HSS.

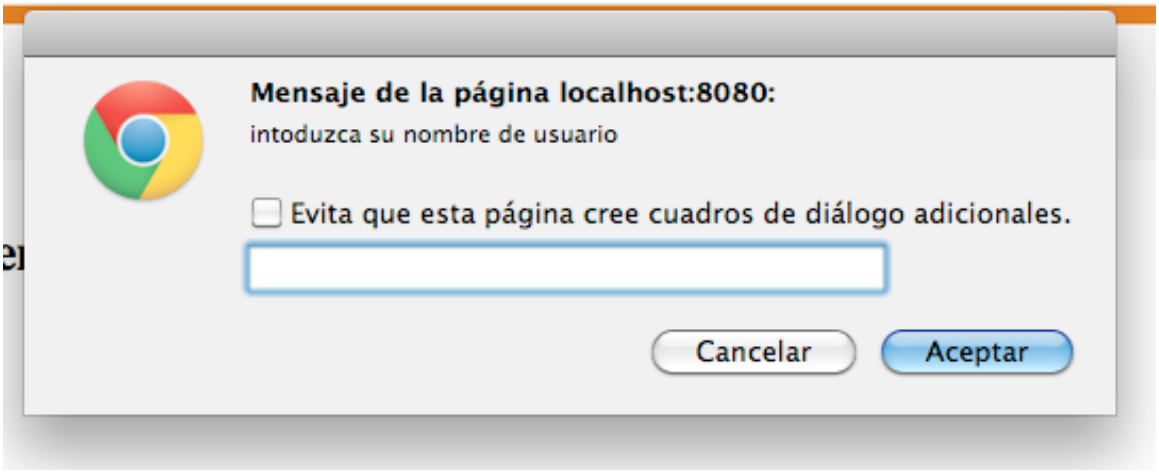


Figura 44: Petición de usuario para registro

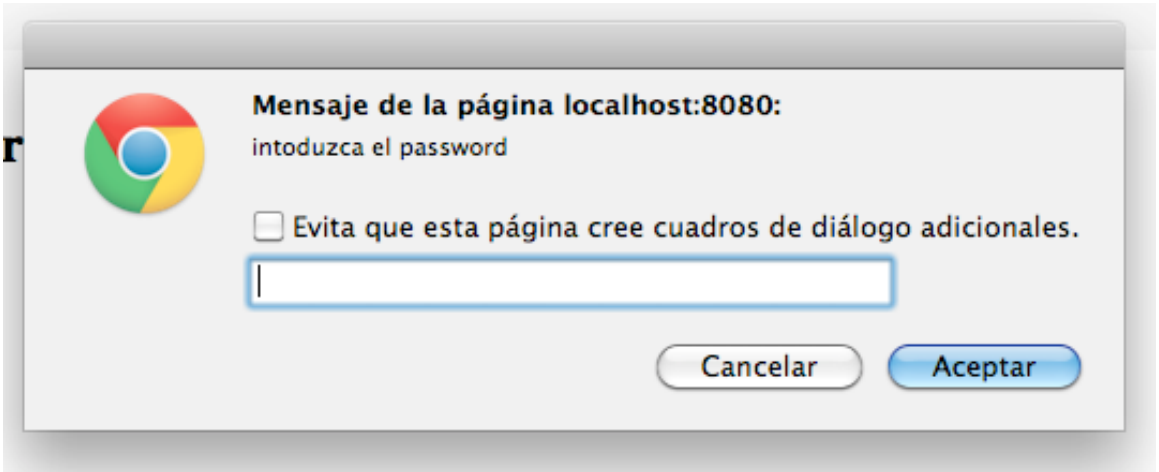


Figura 45: Petición se password para registro

Una vez introducidos estos dos parámetros solicitados, estos datos serán enviados desde el navegador al servidor de la aplicación, el cual ejercerá el papel de agente usuario SIP como se ha comentado con anterioridad e iniciará el registro del usuario. A continuación se muestran los mensajes intercambiados entre la red IMS y el Agente usuario para el registro. Estos mensajes han sido capturados con la herramienta *opensource Wireshark* [33]. Esta herramienta es un analizador de protocolos que permite capturar los mensajes intercambiados en una red y ver su contenido.

1. Primer Mensaje: Petición REGISTER Cuando recibe el usuario y contraseña indicadas por el usuario el Agente Usuario SIP comienza con el registro. El primer mensaje enviado se puede ver a continucación en la información recogida por *Wireshark*.

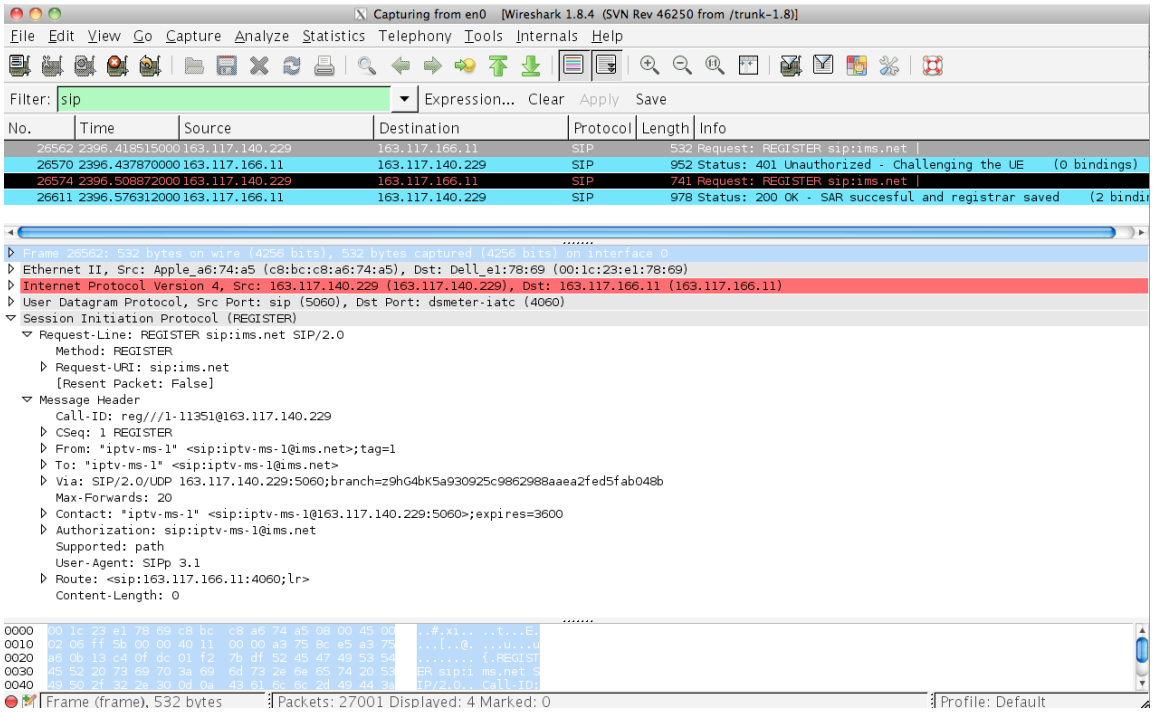


Figura 46: Petición Register

2. Segundo mensaje: Respuesta Unauthorized por parte del HSS. Como se detalló en el partado 3.4.1 de este documento, la primera respuesta por parte del HSS al recibir la primera petición de Registro será una respuesta de No autorización, en la cual se le enviará un reto al usuario.

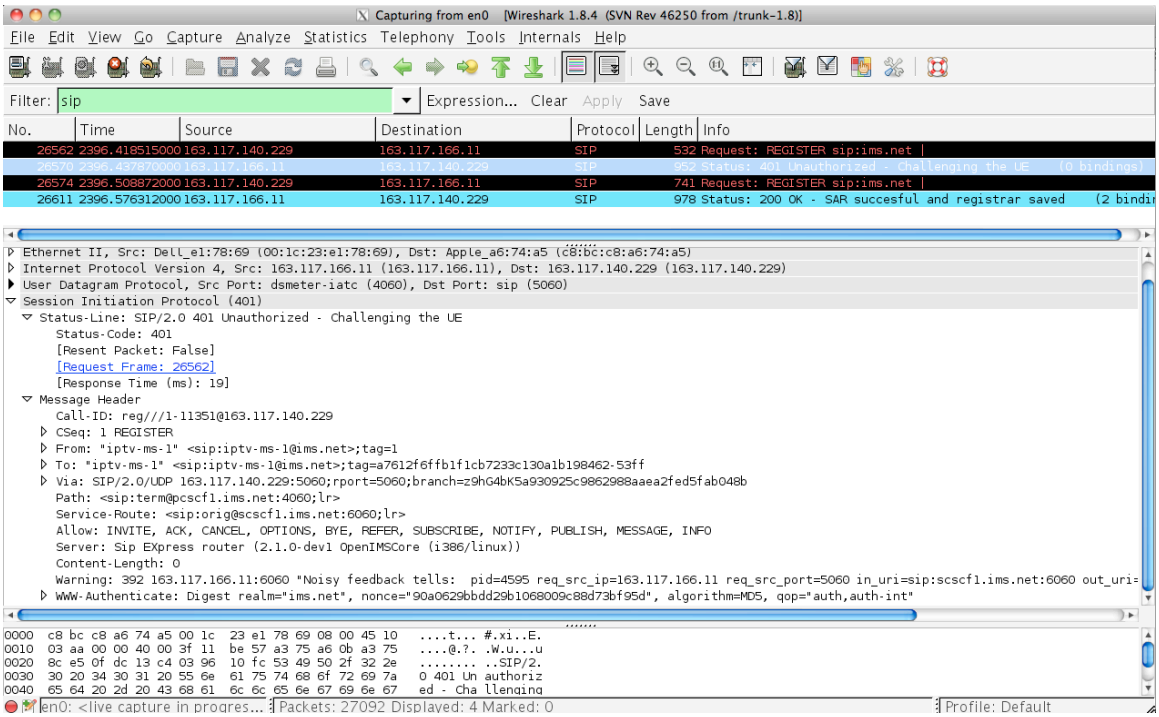


Figura 47: Respuesta Unauthorized

3. Tercer mensaje: Petición de registro con autorización, el agente usuario SIP envía de nuevo el registro con el reto propuesto por el HSS resuelto.

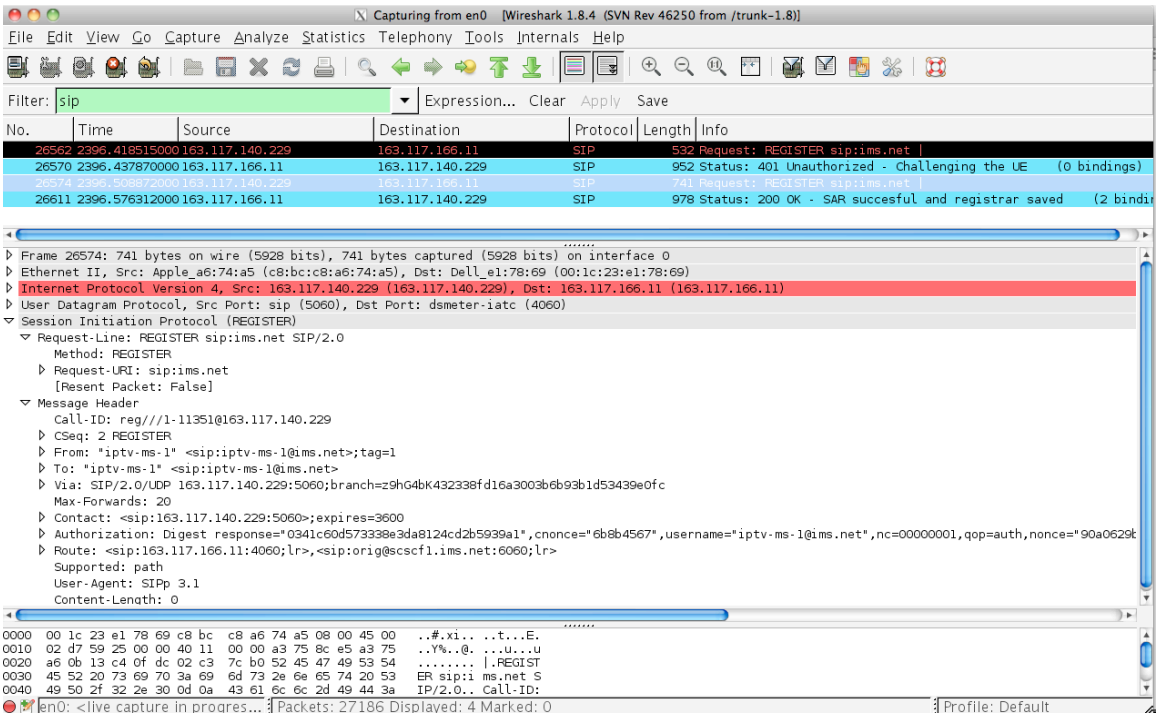


Figura 48: Petición de Registro con autorización

4. Cuarto mensaje: Respuesta 200 OK por parte del HSS. Finalmente el usuario queda registrado de manera exitosa.

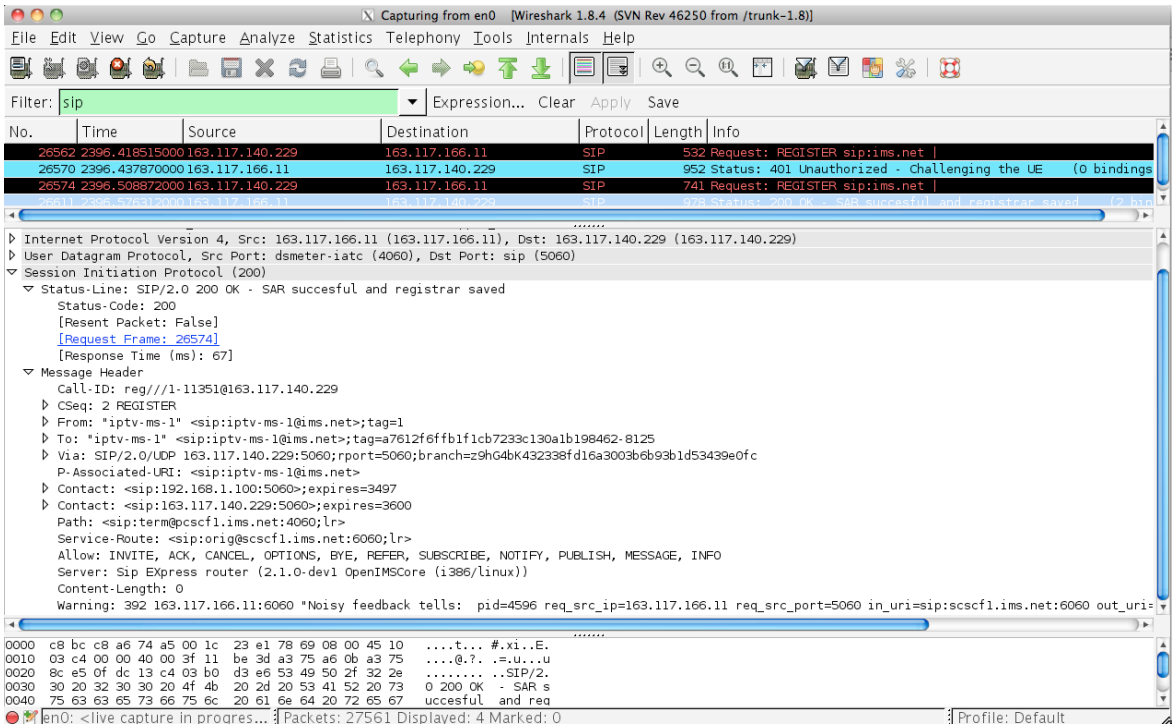
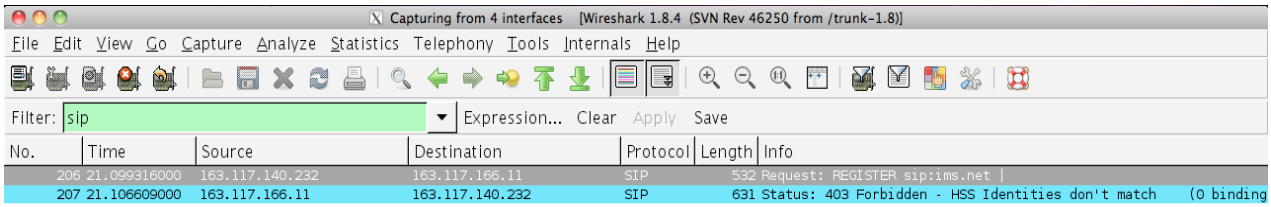


Figura 49: Registro realizado

Como se ha visto el registro ha sido exitoso y se han completado los cuatro pasos descritos en el apartado 3.4. La segunda prueba realizada es el intento de registro de un usuario no registrado en el HSS.



Este registro erróneo se compone sólo de dos mensajes, el mensaje de REGISTER propio y una respuesta por parte de HSS que indica que ese usuario no está dado de alta en el HSS. A continuación se pueden ver los mensajes intercambiados:

```
Request = REGISTER sip:ims.net SIP/2.0
Call-ID: reg///1-11351@163.117.140.232
CSeq: 1 REGISTER
From: "Alexandra" <sip:Alexandra@ims.net>;tag=1
To: "Alexandra" <sip:Alexandra@ims.net>
Via: SIP/2.0/UDP 163.117.140.232:5060;branch=z9hG4bK3cc7775f10b38eaae904b47ff19f7dad
Max-Forwards: 20
Contact: "Alexandra" <sip:Alexandra@163.117.140.232:5060>;expires=3600
Authorization: sip:Alexandra@ims.net
Supported: path
```

User-Agent: SIPp 3.1 Route: <sip:163.117.166.11:4060;lr> Content-Length: 0
Response received with client transaction id gov.nist.javax.sip.stack.SIPClientTransaction@95e1a92: SIP/2.0 403 Forbidden - HSS Identities don't match Call-ID: reg///1-11351@163.117.140.232 CSeq: 1 REGISTER From: "Alexandra" <sip:Alexandra@ims.net>;tag=1 To: "Alexandra" <sip:Alexandra@ims.net>;tag=e927a1ab7c89751999b3b3bdb1ef105c-8413 Via: SIP/2.0/UDP 163.117.140.232:5060;rport=5060;branch=z9hG4bK3cc7775f10b38eaae904b47ff19f7dad Server: Sip EXpress router (2.1.0-dev1 OpenIMSCore (i386/linux)) Warning: 392 163.117.166.11:5060 "Noisy feedback tells: pid=11573 req_src_ip=163.117.166.11 req_src_port=4060 in_uri=sip:ims.net out_uri=sip:ims.net via_cnt==2" Content-Length: 0 Transaction state is Completed Transaction Register failed. Error 403

Con estas dos pruebas ejecutadas comprobamos el correcto funcionamiento del registro de usuarios en la red IMS.

5.3 Acesso al audio y vídeo local

Después del registro de usuarios en la red IMS, el único botón que queda habilitado en la interfaz es el botón de “inicio”. Al presionar esté botón la aplicación nos pedirá la aprobación para poder acceder al audio y vídeo del navegador local.

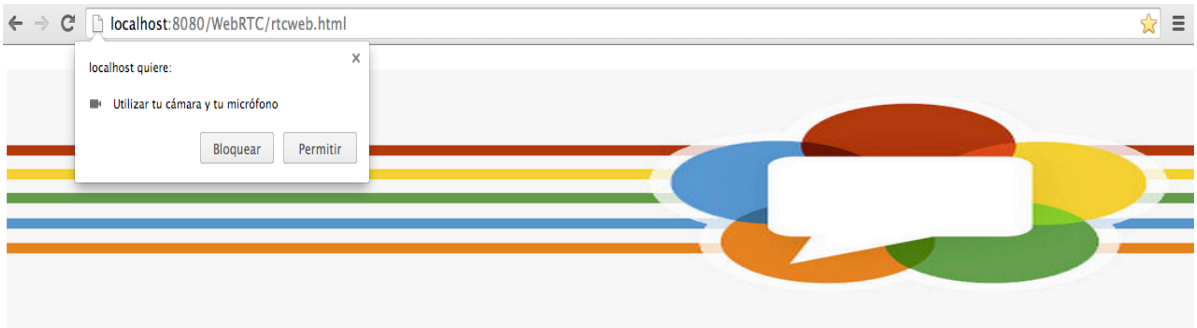


Figura 50: Petición de acceso a audio y vídeo local

Una vez se permita el acesso al audio y vídeo local se nos mostrará en pantalla el fujo que captura nuestro PC.

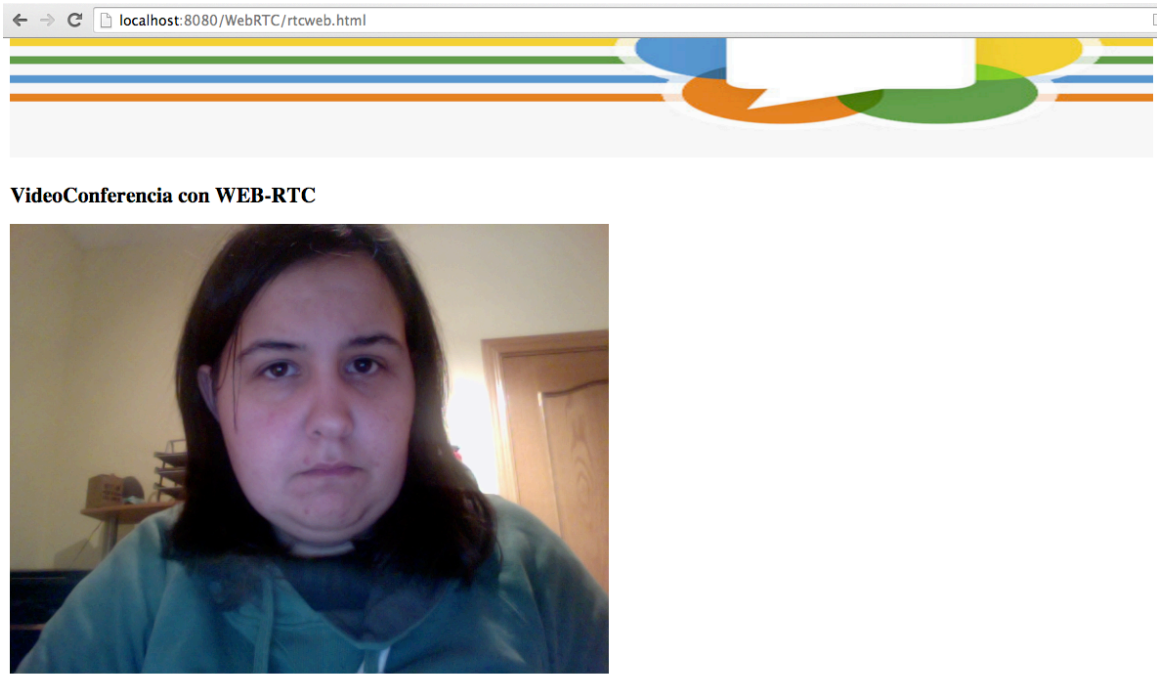


Figura 51: Audio y vídeo local

Con esto queda validado uno de los puntos de seguridad de WebRTC, el usuario debe aceptar el acceso al audio y vídeo local de forma explícita (véase apartado 2.2.4 del presente documento)

5.4 Establecimiento de sesión entre usuarios

Una vez se tiene acesso al audio y vídeo local se puede iniciar la llamada. Para ello se debe presionar el botón "llamar". Esta acción se encarga de varias cosas, por un lado la aplicación nos solicitará el usuario con el que deseamos establecer la comunicación.

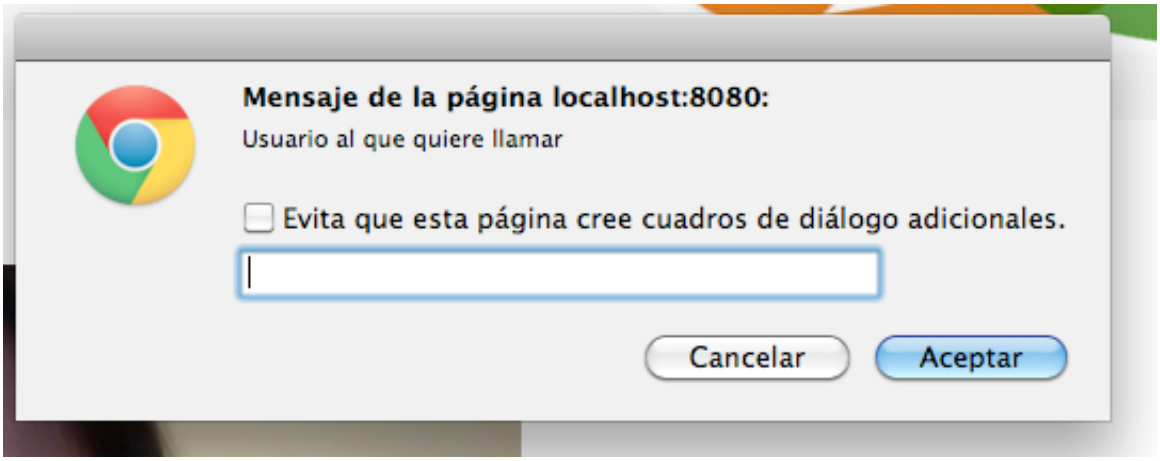


Figura 52: Solicitud de usuario al que se desea llamar

Por otro lado en el momento en el que se presiona el botón “llamar” la aplicación empieza a gestionar el servicio WebRTC. El navegador que inicia la comunicación crea una primera oferta SDP la cual establecerá como descripción local. Esta oferta no es la que se enviará en el mensaje INVITE como carga SDP, la razón es porque esta primera oferta que se genera no tiene los candidatos ICE con la IP y puerto para la conexión.

```
createOfferOnSuccess
type: offer, sdp: v=0
 o=- 2746300836017139575 2 IN IP4 127.0.0.1
 s=-
 t=0 0
 a=group:BUNDLE audio video
 a=msid-semantic: WMS eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
 m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
 c=IN IP4 0.0.0.0
 a=rtcp:9 IN IP4 0.0.0.0
 a=ice-ufrag:JUSa5rxioOgVtq5T
 a=ice-pwd:HGg0WAaeygXnOdNZAXo4K3Qy
 a=fingerprint:sha-256
1C:E2:0D:2E:D1:5F:00:14:38:07:53:5E:81:83:D7:E8:DE:D7:89:D6:D6:A0:4D:FB:B9:7
6:8C:69:1A:13:4C:CA
 a=setup:actpass
 a=mid:audio
 a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
 a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
 a=sendrecv
 a=rtcp-mux
 a=rtpmap:111 opus/48000/2
 a=fmtp:111 minptime=10; useinbandfec=1
 a=rtpmap:103 ISAC/16000
 a=rtpmap:104 ISAC/32000
 a=rtpmap:9 G722/8000
 a=rtpmap:0 PCMU/8000
 a=rtpmap:8 PCMA/8000
 a=rtpmap:106 CN/32000
 a=rtpmap:105 CN/16000
 a=rtpmap:13 CN/8000
……..
```

Como se puede ver la primera oferta que se creo lo hace con una línea “c=IN IP4 0.0.0.0” Esto hará que la conexión sea fallida. Por tanto se debe esperar a que Web-RTC termine de generar todos los candidatos ICE para mandar una oferta correcta.

```
createOfferOnSuccess
type: offer, sdp: v=0
 o=- 2746300836017139575 3 IN IP4 127.0.0.1
 s=-
 t=0 0
 a=group:BUNDLE audio video
 a=msid-semantic: WMS eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
```

```
m=audio 55226 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
c=IN IP4 163.117.140.232
a=rtcp:50209 IN IP4 163.117.140.232
a=candidate:3445025475 1 udp 2122260223 163.117.140.232 55226 typ host generation
0
a=candidate:1229206408 1 udp 2122194687 163.117.219.78 59514 typ host generation
0
a=candidate:3445025475 2 udp 2122260222 163.117.140.232 50209 typ host generation
0
a=candidate:1229206408 2 udp 2122194686 163.117.219.78 64330 typ host generation
0
a=candidate:2211780147 1 tcp 1518280447 163.117.140.232 0 typ host tcptype active
generation 0
a=candidate:130535288 1 tcp 1518214911 163.117.219.78 0 typ host tcptype active
generation 0
a=candidate:2211780147 2 tcp 1518280446 163.117.140.232 0 typ host tcptype active
generation 0
a=candidate:130535288 2 tcp 1518214910 163.117.219.78 0 typ host tcptype active
generation 0
a=ice-ufrag:JUSa5rxioOgVtq5T
a=ice-pwd:HGg0WAaeygXnOdNZAXo4K3Qy
a=fingerprint:sha-256
1C:E2:0D:2E:D1:5F:00:14:38:07:53:5E:81:83:D7:E8:DE:D7:89:D6:D6:A0:4D:FB:B9:7
6:8C:69:1A:13:4C:CA
a=setup:actpass
a=mid:audio
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=sendrecv
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10; useinbandfec=1
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:9 G722/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:126 telephone-event/8000
a=maxptime:60
……
```

Esta segunda oferta contiene toda la información necesaria para poder establecer la comunicación. Esta oferta se enviará al Agente Usuario SIP para que forme la petición INVITE e iniciar el proceso de señalización. La petición INVITE atravesará el CORE IMS y llegará al segundo Agente usuario SIP que pasará la carga SDP recibida en la petición a su navegador. Dicho navegador establecerá la carga SDP recibida como su descripción remota y generará una respuesta con sus CODECs y capacidades propias. Al

igual que en el caso de la oferta se generará una primera respuesta que se establecerá como descripción local.

```
createAnswerOnSuccess
type: answer, sdp: v=0
o=- 5413457662029002875 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:CJW4F3C9ipS7H8bm
a=ice-pwd:+zZV7IAaycebJRDDrg/ngOaX
a=fingerprint:sha-256
F0:8D:79:77:1D:4B:04:E7:C0:5A:2C:D3:92:FF:86:03:C6:AE:7F:5A:8C:94:7E:DA:59:8
8:53:51:18:D0:66:53
a=setup:active
a=mid:audio
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=sendrecv
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10; useinbandfec=1
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:9 G722/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:126 telephone-event/8000
a=maxptime:60
a=ssrc:3227282109 cname:nLxldIqLF7ZupfWq
a=ssrc:3227282109 msid:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd 971137af-
520e-45a4-b609-7497d3c220db
a=ssrc:3227282109 mslabel:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
a=ssrc:3227282109 label:971137af-520e-45a4-b609-7497d3c220db
……
```

Esta primera respuesta no se enviará en el mensaje de OK ya que al igual que en la primera oferta contine la siguiente línea “c=IN IP4 0.0.0.0” que provocaría un fallo en la comunicación. De nuevo hay que esperar a que se generen todos los candidatos ICE para generar una nueva respuesta.

```
createAnswerOnSuccess
type: answer, sdp: v=0
o=- 5413457662029002875 3 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
m=audio 50801 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
c=IN IP4 163.117.140.22
a=rtcp:9 IN IP4 0.0.0.0
a=candidate:183249902 1 udp 2122194687 163.117.140.22 50801 typ host generation 0
a=candidate:4136782018 1 udp 2122262783 2001:720:410:1020:74af:532:282c:27b0
34390 typ host generation 0
a=candidate:1474200709 1 udp 2122063615 163.117.223.127 49861 typ host generation
0
a=candidate:725943426 1 udp 2122131711 2001:720:410:3343:e0e8:d9bc:f23:442b
54283 typ host generation 0
a=candidate:1148180254 1 tcp 1518214911 163.117.140.22 0 typ host tcptype active
generation 0
a=candidate:3088167986 1 tcp 1518283007 2001:720:410:1020:74af:532:282c:27b0 0
typ host tcptype active generation 0
a=candidate:425556085 1 tcp 1518083839 163.117.223.127 0 typ host tcptype active
generation 0
a=candidate:1707538546 1 tcp 1518151935 2001:720:410:3343:e0e8:d9bc:f23:442b 0
typ host tcptype active generation 0
a=ice-ufrag:CJW4F3C9ipS7H8bm
a=ice-pwd:+zZV7IAaycebJRDDrg/ngOaX
a=fingerprint:sha-256
F0:8D:79:77:1D:4B:04:E7:C0:5A:2C:D3:92:FF:86:03:C6:AE:7F:5A:8C:94:7E:DA:59:8
8:53:51:18:D0:66:53
a=setup:active
a=mid:audio
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=sendrecv
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10; useinbandfec=1
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:9 G722/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:126 telephone-event/8000
a=maxptime:60
a=ssrc:3227282109 cname:nLxldIqLF7ZupfWq
a=ssrc:3227282109 msid:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd 971137af-
```

```
520e-45a4-b609-7497d3c220db
a=ssrc:3227282109 mslabel:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
a=ssrc:3227282109 label:971137af-520e-45a4-b609-7497d3c220db
…….
```

Esta descripción SDP, ya completa, será pasada al Agente usuario SIP que la encapsulará en un mensaje OK el cual mandará a través de la red IMS hasta el otro extremo de la comunicación. Une vez sea recibido por el Agente usuario SIP, éste pasará la carga SDP recibida a su navegador el cual establecerá dicha descripción como descripción remota. En este momento los dos extremos de la comunicación ya conocen todos los datos necesarios para poder iniciar la comunicación y la videollamada queda establecida. En el Anexo C se pueden ver todas las descripciones SDP completas que han sido generadas por los dos extremos de la comunicación.

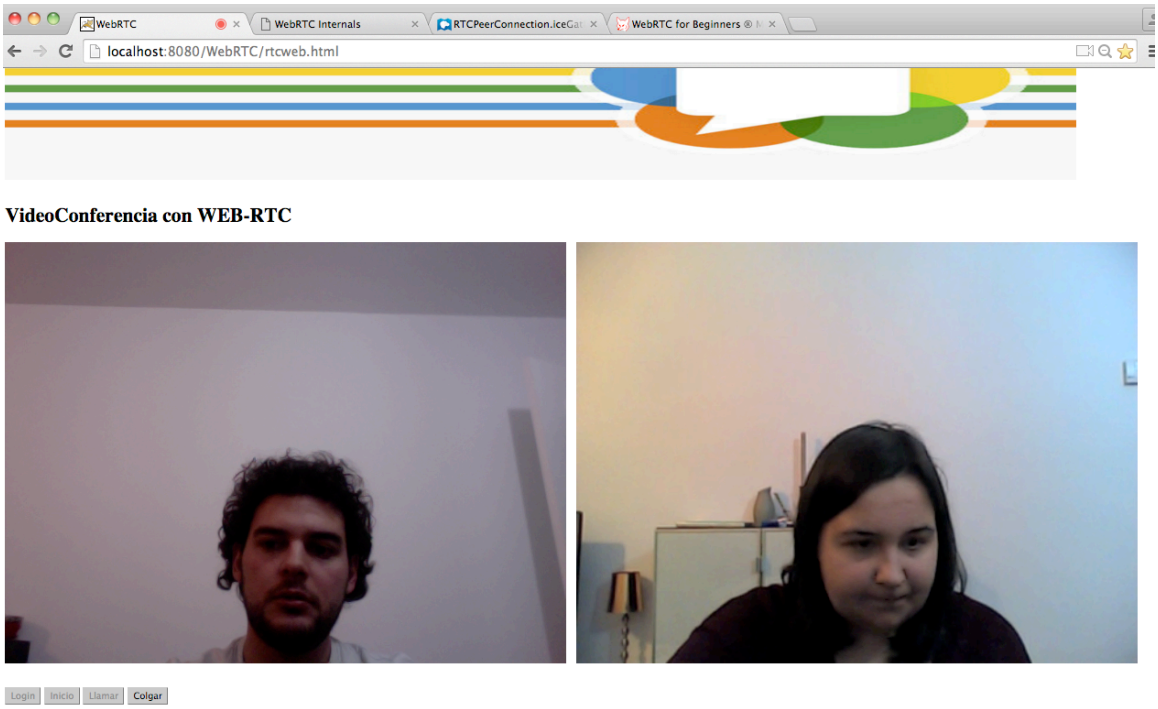


Figura 53: Vídeo llamada establecida

En la imagen anterior podemos ver como queda establecida la comunicación entre los dos usuarios finales.

Google Chrome ha desarrollado una herramienta interna para Web-RTC con la que poder hacer seguimiento de todo lo ocurrido en el establecimiento de la conexión entre usuarios. Dicha herramienta es accesible mediante la web *chrome://webrtc-internals/,* y en ella podemos ver todo el proceso seguido para las descripciones locales y remotas, como se generan los candidatos de cada navegador y como se transmite y recibe el audio y vídeo de ambos navegadores. A continuación se pueden ver algunos ejemplos de la información que muestra esta herramienta web propia de Google.

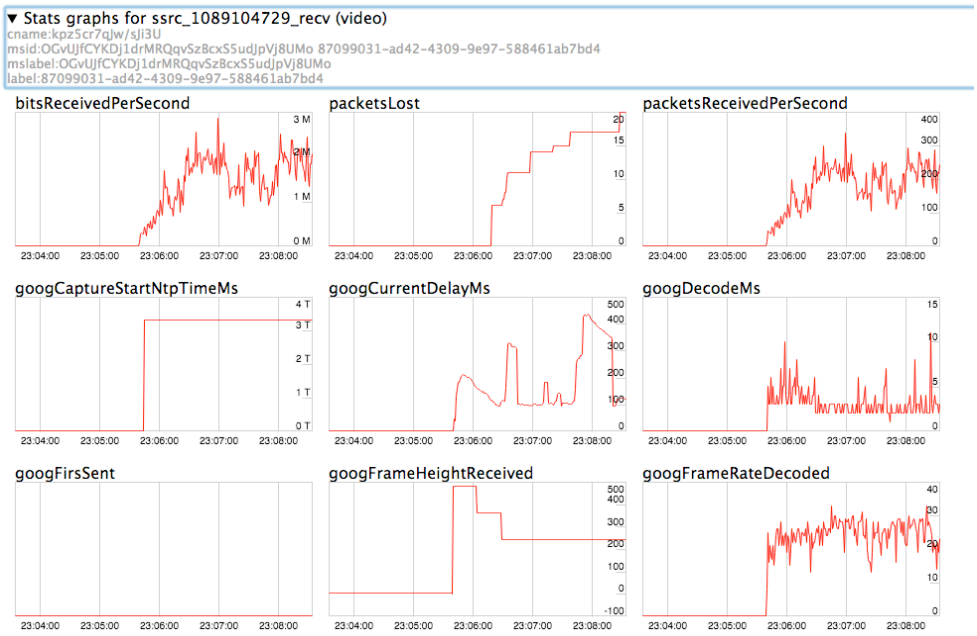
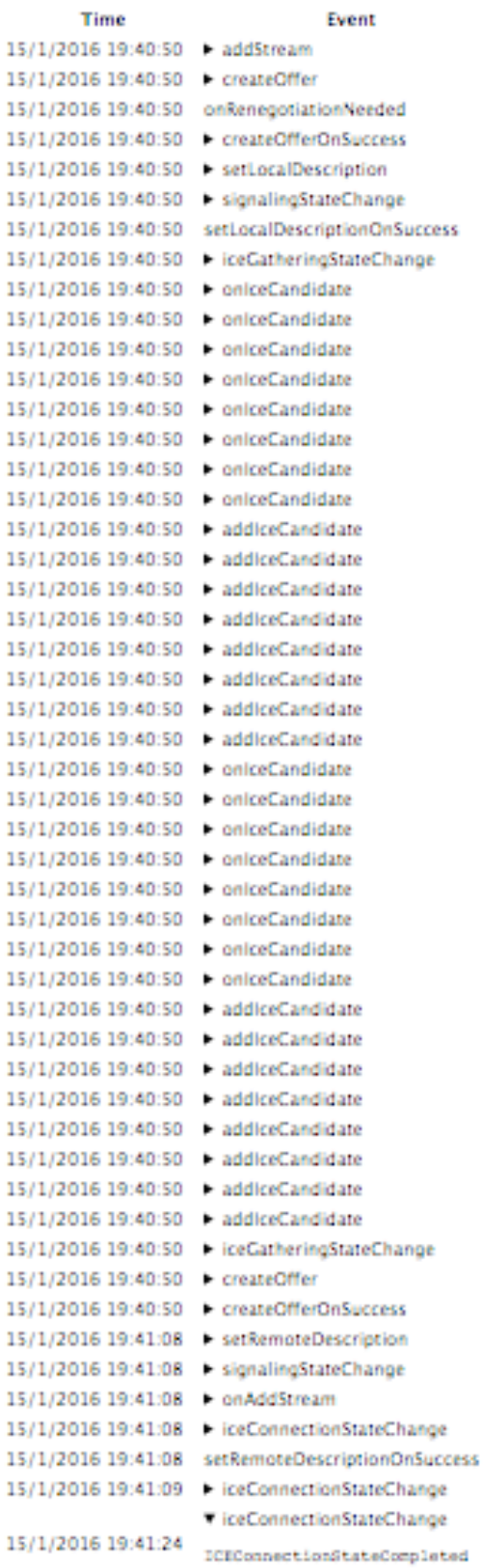


Figura 54: Gráficas de vídeo recibido

Figura 55: Gráficas de audio recibido



Time	Event
15/1/2016 19:40:50	▸ addStream
15/1/2016 19:40:50	▸ createOffer
15/1/2016 19:40:50	onRenegotiationNeeded
15/1/2016 19:40:50	▸ createOfferOnSuccess
15/1/2016 19:40:50	▸ setLocalDescription
15/1/2016 19:40:50	▸ signalingStateChange
15/1/2016 19:40:50	setLocalDescriptionOnSuccess
15/1/2016 19:40:50	▸ iceGatheringStateChange
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ onIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ addIceCandidate
15/1/2016 19:40:50	▸ iceGatheringStateChange
15/1/2016 19:40:50	▸ createOffer
15/1/2016 19:40:50	▸ createOfferOnSuccess
15/1/2016 19:41:08	▸ setRemoteDescription
15/1/2016 19:41:08	▸ signalingStateChange
15/1/2016 19:41:08	▸ onAddStream
15/1/2016 19:41:08	▸ iceConnectionStateChange
15/1/2016 19:41:08	setRemoteDescriptionOnSuccess
15/1/2016 19:41:09	▸ iceConnectionStateChange
15/1/2016 19:41:24	▾ iceConnectionStateChange ICEConnectionStateCompleted

Figura 56: Establecimiento de descripciones

Capítulo 6

Conclusiones y líneas futuras

6.1 Conclusiones

Este Proyecto Fin de Carrera ha consistido en el desarrollo e implementación de una aplicación de videollamada basada en SIP y WebRTC. La aplicación se ha desarrollado para que sea acessible vía web, desde un navegador Google Chrome con una versión 29 o superior y de forma que sea compatible con un plano de control IMS. El diseño del sistema se ha estructurado en dos planos diferenciados, el primero corresponde al plano de usuario que se encargará de la interacción del usuario con la aplicación, por tanto este plano sólo se encuentra en la aplicación de cliente. El segundo plano corrresponde al plano de control, encargado del registro de los usuarios en la red IMS, de la gestión de la señalización para el establecimiento de sesión entre los usuarios finales y de la gestión de la funcionalidad aportada por WebRTC, por tanto este plano lo encontraremos dividido entre la aplicación de cliente y el servidor web. Dentro de estos dos planos tenemos 2 componentes distintos:

- Aplicacción de cliente: Por una parte implemente el plano de usuario que permite, mediante la interfaz gráfica, la interación del usuario con la aplicación, le permite registrarse en la red IMS indicando un nombre y una contraseña váilidos, le permite indicar el nombre del usuario con el que desea establecer la comunicación y le permite autorizar el acceso al audio y vídeo

local para el uso de WebRTC. Por otro lado implemente también parte del plano de control, el cual le permite la correcta gestión de las funcionalidades de WebRTC. Este plano será el encargado de gestionar tanto las descripciones locales y remotas para el correcto funcionamiento de WebRTC, así como de generar las ofertas y respuestas que sean necesarias para el establecimiento de sesión. Por último se encargará de capturar y transmitir el audio y vídeo local y de mostrar el audio y vídeo del otro lado de la comunicación.

- Servidor web: Implementa sólo el plano de control y hará la función de agente usuario SIP. Es el encargado de la gestión de todos los mensajes correspondientes a la parte de la señalización del sistema. Se encarga del registro del usuario en el Core IMS, así como de la correcta gestión de los mensajes necesarios para el establecimiento de sesión con el otro extremo de la comunicación.

Aunque Web-RTC es una tecnología que está cobrando cada vez más importancia dentro del mundo de las telecomunicaciones aún está poco explotada y tiene un potencial muy elevado para el desarrollo de aplicaciones en la web. Como se explicó en apartados anteriores de esta memoria, la principal ventaja con la que cuenta es que es una tecnología desarrollada exclusivamente para el mundo Web. Esto facilita a los desarrolladores la implementación de aplicaciones o complementos utilizando esta tecnología. Como queda demostrado con este Proyecto Fin de Carrera esta tecnologíaa emergente aporta los resultados esperados y es una manera sencilla de establecer comunicaciones multimedia entre navegadores. El principal inconveniente con el que cuenta es que al ser una tecnología poco explotada tiene una documentación limitada y esto no facilita la resolución de los problemas que van apareciendoa la hora de ir implementando aplicaciones.Así mismo cabe recordar que WebRTC aún no está implementado en todos los navegadores [1], por lo que su uso se ve limitado debido a este inconveniente.

6.2 Futuras líneas de trabajo

Tras la finalización del trabajo propuesto en el inicio, consistente en el desarrollo de una aplicación de vídeo llamada basada en SIP y web-RTC, se proponen futuras líneas de trabajo para la mejora de la aplicación desarrollada y para la introducción de nuevas funcionalidades. Estas futuras líneas de trabajo son:

- Adaptación de la aplicación desarrollada para diferentes tipos de navegadores. La aplicación se ha desarrollado exclusivamente para ser ejecutada en un navegador Google Chrome, ya que dicho navegador fue uno primeros en soportar Web-RTC. En la actualidad Web-RTC está soportado por los navegadores Google chrome, Mozilla firefox y Ópera. Aún no tiene soporte para Internet explorer y Safari [24]. Esta adaptación permitiría ejecutar vídeo llamadas entre navegadores de distinto tipo.

- Adaptación de la aplicación desarrollada para dispositivos móviles. En la actualidad el uso de dispositivos móviles está muy extendido. Se propone adaptar la aplicación para que pueda ser utilizada en smartphones o tablets, esto es posible ya que únicamente se debe contar con un navegador que sea capaz de utilizar la tecnología Web-RTC y se deberá adaptar la interfaz gráfica a dichos dispositivos.

- Adaptación de la aplicación desarrollada para vídeo llamadas de conferencia. Se propone el estudio e implementación de un sistema que permita la llamada entre mas de dos usuarios. Este tipo de aplicación sería muy útil tanto en el mundo empresarial como para el resto de usuarios.

- Implementación de servicios complementarios dentro de la aplicación. Se propone el desarrollo de varios complementos a la aplicación desarrollada , incluir un servicio de compartición de archivos, incluir un servicio de presencia y un servicio de mensajería instantánea. Se deberán estudiar las capacidades de Web-RTC para el desarrollo de esta funcionalidad complementaria.

- Se propone una línea de investigación en la que se estudie la posibilidad de que los propios navegadores web sean capaces de soportar la señalización necesaria para el establecimiento de sesiones.

Capítulo 7

Anexos

Anexo A: Presupuesto

En este anexo se detallan las tareas realizadas para el desarrollo de este proyecto fin de carrera, así como el tiempo empleado en completar cada una de ellas y los recursos utilizados.

- **Tarea 1: Definición de objetivos**

  Esta tarea se basa en la definición de los objetivos que se quieren conseguir con el desarrollo de este proyecto. Se quiere desarrollar un sistema de videollamada basado en SIP y WebRTC. Para su correcto desarrollo es necesario estudiar los protocolos que se emplearán y las herramientas necesarias que deben ser utilizadas

  - Plano de control: Se utilizará el protocolo SIP para la gestión de la señalización. El protocolo SDP especificará el detalle de la sesión y esta señalización pasará a través del Core-IMS del laboratio de Ingeniería Telemática de la universidad Carlos III de Madrid. Para toda la implementación se decide utilizar los siguientes lenguajes de programación: Java, HTML, JavaScript y XML.
    Duración: 3 semanas

  - Plano de usuario: La gestión de los datos multimedia se harán con el protocolo RTP y la ejecución del proyecto se hará sobre un navegador Google Chrome.
    Duración: 2 semanas

- **Tarea 2: Desarrollo del Estado del arte**

  Una vez establecidos los objetivos que tendrá el desarrollo de este proyecto es necesario estudiar y comprender cada uno de los protocolos y herramientas a utilizar

  - WebRTC: Es necesario comprender el funcionamiento y la definición de esta tecnología emergente para el desarrollo del proyecto, ya que será su principal componente.
    Duración: 2,5 semanas

  - IMS: Estudio de la arquitectura y funcionamiento de la red.
    Duración: 2 semanas

  - SIP: Estudio del protocolo SIP para el establecimiento de sesiones entre usuarios.
    Duración: 2 semanas

  - SDP y RTP: Estudio de los campos SDP para el establecimiento de una sesión multimedia y RTP para la gestión de esta sesión.
    Duración: 1 semana

- Lenguajes de programación: Conocimiento de los lenguajes de programación que serán utilizados, así como el funcionamiento de Servlet y tecnología Ajax. Es necesario conocer también las APIS encargadas de la gestión del protocolo SIP en java, JAIN-SIP. Duración: 2 semanas.

- **Tarea 3: Identificación de los requisitos del sistema**

  Una vez terminado el estudio de todos los protocolos que serán utilizados y con el conocimiento de los lenguajes de programación que serán necesarios hay que definir cómo se va a implementar el sistema. Se especifica de forma detallada los requisitos que debe cumplir cada parte de la implementación

  - Estudio de los requisitos en el plano de control
    Duración: 1,5 Semanas

  - Estudio de los requisisitos en el plano de usuario
    Duración: 1,5 Semanas

- **Tarea 4: Implementación del sistema**

  Una vez Identificados los requisitos que debe cumplir el sistema, se pasa a su implementación:

  - Servidor web: Desarrollo de la gestión de la señalización para el registro de usuarios en la red IMS y el establecimiento de sesión entre navegadores.
    Duración: 3 semanas

  - Aplicación de cliente: Desarrollo de la interfaz gráfica y el servlet que utiliza, así como la funcionalidad WebRTC que se utiliza.
    Duración: 3 semanas

- **Tarea 5: Pruebas**

  Una vez completado el desarrollo de la implementación del sistema por completo se debe validar dicha implementación con las pruebas oportunas. La prueba realizada será el establecimiento de una video llamada entre dos navegadores. Esta vídeo llamada será analizada en 4 pasos para validar por complemto el desarrollo.

  - Registro de usuarios en la red IMS.: Se prueba a registrar un usuario correctamente dado de alta en el HSS y un usuario no dado de alta en el HSS para contrastar datos.
    Duración 1’5 semana

  - Acceso al audio y vídeo local: Se comprueba el correcto acceso al flujo multimedia del usuario local.
    Duración 1 semana

- Establecimiento de sesión entre 2 navegadores: Se prueba la correcta transmisión y recepción de audio y vídea entre los dos navegadores. Duración: 2`5 semanas

Una vez descritas todas las tareas realizadas para el desarrollo de este proyecto se obtienen un total de 28,5 semanas de trabajo. Queda reflejado en la siguiente tabla

Tarea	**Duración**	**Recurso**
Definición de objetivos	5 Semanas	1 ingeniero/ 4h diarias
Desarrollo del Estado del arte	9,5 Semanas	1 ingeniero/ 4h diarias
Identificación de los requisitos del sistema	3 Semanas	1 ingeniero/ 4h diarias
Implementación del sistema	6 Semanas	1 ingeniero/ 4h diarias
Pruebas	5 Semanas	1 ingeniero/ 4h diarias
Total	28,5 Semanas	570 horas

Tabla 9: Tareas para el desarrollo del proyecto

Se considera que que cada semana tiene 5 días hábiles de trabajo y se dedican 4 horas diarias al desarrollo del proyecto. En total serán necesarias un total de 28,5 semanas, lo que equivalen a 570 horas de trabajo, un total de 7 meses de trabajo aproximadamente.

No se tienen en cuenta para los cálculos de este presupuesto los costes materiales de los portátiles que se han empleado para la implementación y pruebas, ya que suponemos que sólo se requiere el desarrollo de la aplicación. Teniendo esto en cuenta, y según los datos del COIT (Colegio Oficial de Ingenieros de Telecomunicaciones) [25] en su informe “El ingeniero de telecomunicaciones: Perfil Socio-Profesional”, en el cual se detalla que una mújer ingeniera con menos de 5 años de experiencia tiene un salario medio anual de 24.406,2€ podemos concluir que el coste total del proyecto son 14.236,95€

Personal	**Bruto Anual(€)**	**Total Meses**	**Total(€)**
1 Ingeniero	24.406,2	7	14.236,95

Tabla 10: Presupuesto del desarrollo

Anexo B: Atributos opcionales en sesión SDP

En este anexo se describen los campos opcionales en las sesiones SDP

Session Information (“i=”)

i= <session description >

El campo Session Information proporciona información adicional sobre la sesión.

URI (“u=”)

u=<uri>

El campo URI es un identificador similar al utilizado en la web. Es un puntero a información adicional sobre la sesión. No es un campo obligatorio pero en el caso de estar presente en la sesión debe ir antes del primer campo media (“m=”). Sólo está permitido un campo URI por sesión.

Email Address and Phone number (“e=” y “p=”)

e=<email-address>
p=<phone-number>

El campo email-address contiene la dirección de correo electrónico de la persona responsable del establecimiento de la sesión.

El campo phone-number contiene el teléfono de contacto de la persona responsable del establecimiento de la sesión

Bandwidth (“b=”)

b= <bwtype> : <bandwidth>

Se trata de un campo que almacena el ancho de banda propuesto para utilizar en la sesión durante la transmisión de datos.

El bwtype puede ser CT cuando se tienen varias sesiones en paralelo de manera que indica el ancho de banda total que se puede usar por todos los participantes de la sesión o AS cuando se refiere al ancho de banda utilizado por una única sesión.
El parámetro bandwidth es el valor numérico expresado en kilobytes por segundo.

Repeat times (“r=”)

r= <repeat interval> <active duration> <offsets from start-time>

Esta línea representa cada cuanto se tiene que repetir una determinada sesión.

- repeat interval: indica cada cuanto tiempo se debe repetir la sesión

- active duration: indica durante cuánto tiempo se debe repetir la sesión
- offsets from start-time: en caso de que haya mas de una sesión activa

Time Zones (“z=”)

z= <adjustment time> <offset>

Se utiliza para hacer ajustes de tiempo de diferentes zonas horarias.

Encryption keys (“k=”)

k= <method>
k= <method> : <encryption key>

La línea k contiene la clave utilizada para la encriptación de los datos.

El método puede ser clear, base64, uri, o prompt. Si es prompt, la clave no se especifica en el parámetro encryption key, en cualquiera de los otros casos la clave se especifica en este parámetro.

Attributes (“a=”)

a= <attribute>
a= <attribute> : <value>

Los atributos son el medio principal para extender el protocolo SDP. Pueden definirse atributos a nivel de sesión y a nivel de media.

El campo atributo tiene dos formatos distintos:

- a= <attribute> son los llamados atributos de propiedad. Son atributos a nivel de sesión.
- a= <attribute> : <value>

Anexo C: Ofertas y Respuestas SDP

En este Anexo se detallan todas las Ofertas y Respuestas SDP que se generan dentro de WebRTC. Es interesante comprobar que los CODECs de audio y vídeo intercambiados son los propios de WebRTC. Estos CODECs son los descritos en el apartado 2.2.2 del presente documento.

Primera Oferta generada por parte del primer navegador, del usuario p2pIPTV. Esta primera oferta será lo que es usuario p2pIPTV establezca como descripción local. Se marca en rojo lo más destacado de dicha oferta

```
createOfferOnSuccess
type: offer, sdp: v=0
 o=- 2746300836017139575 2 IN IP4 127.0.0.1
 s=-
 t=0 0
 a=group:BUNDLE audio video
 a=msid-semantic: WMS eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
 m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
 c=IN IP4 0.0.0.0
 a=rtcp:9 IN IP4 0.0.0.0
 a=ice-ufrag:JUSa5rxioOgVtq5T
 a=ice-pwd:HGg0WAaeygXnOdNZAXo4K3Qy
 a=fingerprint:sha-256
1C:E2:0D:2E:D1:5F:00:14:38:07:53:5E:81:83:D7:E8:DE:D7:89:D6:D6:A0:4D:FB:B9:7
6:8C:69:1A:13:4C:CA
 a=setup:actpass
 a=mid:audio
 a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
 a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
 a=sendrecv
 a=rtcp-mux
 a=rtpmap:111 opus/48000/2
 a=fmtp:111 minptime=10; useinbandfec=1
 a=rtpmap:103 ISAC/16000
 a=rtpmap:104 ISAC/32000
 a=rtpmap:9 G722/8000
 a=rtpmap:0 PCMU/8000
 a=rtpmap:8 PCMA/8000
 a=rtpmap:106 CN/32000
 a=rtpmap:105 CN/16000
 a=rtpmap:13 CN/8000
 a=rtpmap:126 telephone-event/8000
 a=maxptime:60
 a=ssrc:2662311244 cname:9vOFkQsNuajNvEpF
 a=ssrc:2662311244 msid:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7 ea7ece43-
203b-4405-a904-5a3c44b4c9a5
 a=ssrc:2662311244 mslabel:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
 a=ssrc:2662311244 label:ea7ece43-203b-4405-a904-5a3c44b4c9a5
```

```
m=video 9 UDP/TLS/RTP/SAVPF 100 116 117 96
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:JUSa5rxioOgVtq5T
a=ice-pwd:HGg0WAaeygXnOdNZAXo4K3Qy
a=fingerprint:sha-256
1C:E2:0D:2E:D1:5F:00:14:38:07:53:5E:81:83:D7:E8:DE:D7:89:D6:D6:A0:4D:FB:B9:7
6:8C:69:1A:13:4C:CA
a=setup:actpass
a=mid:video
a=extmap:2 urn:ietf:params:rtp-hdrext:toffset
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:4 urn:3gpp:video-orientation
a=sendrecv
a=rtcp-mux
a=rtpmap:100 VP8/90000
a=rtcp-fb:100 ccm fir
a=rtcp-fb:100 nack
a=rtcp-fb:100 nack pli
a=rtcp-fb:100 goog-remb
a=rtpmap:116 red/90000
a=rtpmap:117 ulpfec/90000
a=rtpmap:96 rtx/90000
a=fmtp:96 apt=100
a=ssrc-group:FID 194275495 2065741749
a=ssrc:194275495 cname:9vOFkQsNuajNvEpF
a=ssrc:194275495 msid:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7 f32658ea-37dc-
46e1-81ef-7b8ae53bd536
a=ssrc:194275495 mslabel:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
a=ssrc:194275495 label:f32658ea-37dc-46e1-81ef-7b8ae53bd536
a=ssrc:2065741749 cname:9vOFkQsNuajNvEpF
a=ssrc:2065741749 msid:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7 f32658ea-37dc-
46e1-81ef-7b8ae53bd536
a=ssrc:2065741749 mslabel:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
a=ssrc:2065741749 label:f32658ea-37dc-46e1-81ef-7b8ae53bd536
```

Segunda oferta generada con los candidatos ICE. Dicha oferta será la que se mande en el mensaje INVITE. Esta oferta SDP será la que el segundo usuario, bob, establezca como descripción remota. Se marca en rojo lo más significativo de dicha oferta.

```
createOfferOnSuccess
type: offer, sdp: v=0
o=- 2746300836017139575 3 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
m=audio 55226 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
c=IN IP4 163.117.140.232
a=rtcp:50209 IN IP4 163.117.140.232
```

```
 a=candidate:3445025475 1 udp 2122260223 163.117.140.232 55226 typ host generation
0
 a=candidate:1229206408 1 udp 2122194687 163.117.219.78 59514 typ host generation
0
 a=candidate:3445025475 2 udp 2122260222 163.117.140.232 50209 typ host generation
0
 a=candidate:1229206408 2 udp 2122194686 163.117.219.78 64330 typ host generation
0
 a=candidate:2211780147 1 tcp 1518280447 163.117.140.232 0 typ host tcptype active
generation 0
 a=candidate:130535288 1 tcp 1518214911 163.117.219.78 0 typ host tcptype active
generation 0
 a=candidate:2211780147 2 tcp 1518280446 163.117.140.232 0 typ host tcptype active
generation 0
 a=candidate:130535288 2 tcp 1518214910 163.117.219.78 0 typ host tcptype active
generation 0
 a=ice-ufrag:JUSa5rxioOgVtq5T
 a=ice-pwd:HGg0WAaeygXnOdNZAXo4K3Qy
 a=fingerprint:sha-256
1C:E2:0D:2E:D1:5F:00:14:38:07:53:5E:81:83:D7:E8:DE:D7:89:D6:D6:A0:4D:FB:B9:7
6:8C:69:1A:13:4C:CA
 a=setup:actpass
 a=mid:audio
 a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
 a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
 a=sendrecv
 a=rtcp-mux
 a=rtpmap:111 opus/48000/2
 a=fmtp:111 minptime=10; useinbandfec=1
 a=rtpmap:103 ISAC/16000
 a=rtpmap:104 ISAC/32000
 a=rtpmap:9 G722/8000
 a=rtpmap:0 PCMU/8000
 a=rtpmap:8 PCMA/8000
 a=rtpmap:106 CN/32000
 a=rtpmap:105 CN/16000
 a=rtpmap:13 CN/8000
 a=rtpmap:126 telephone-event/8000
 a=maxptime:60
 a=ssrc:2662311244 cname:9vOFkQsNuajNvEpF
 a=ssrc:2662311244 msid:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7 ea7ece43-
203b-4405-a904-5a3c44b4c9a5
 a=ssrc:2662311244 mslabel:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
 a=ssrc:2662311244 label:ea7ece43-203b-4405-a904-5a3c44b4c9a5
 m=video 49689 UDP/TLS/RTP/SAVPF 100 116 117 96
 c=IN IP4 163.117.140.232
 a=rtcp:56955 IN IP4 163.117.140.232
 a=candidate:3445025475 1 udp 2122260223 163.117.140.232 49689 typ host generation
0
 a=candidate:1229206408 1 udp 2122194687 163.117.219.78 53868 typ host generation
```

```
0
 a=candidate:3445025475 2 udp 2122260222 163.117.140.232 56955 typ host generation
0
 a=candidate:1229206408 2 udp 2122194686 163.117.219.78 50443 typ host generation
0
 a=candidate:2211780147 1 tcp 1518280447 163.117.140.232 0 typ host tcptype active
generation 0
 a=candidate:130535288 1 tcp 1518214911 163.117.219.78 0 typ host tcptype active
generation 0
 a=candidate:2211780147 2 tcp 1518280446 163.117.140.232 0 typ host tcptype active
generation 0
 a=candidate:130535288 2 tcp 1518214910 163.117.219.78 0 typ host tcptype active
generation 0
 a=ice-ufrag:JUSa5rxioOgVtq5T
 a=ice-pwd:HGg0WAaeygXnOdNZAXo4K3Qy
 a=fingerprint:sha-256
1C:E2:0D:2E:D1:5F:00:14:38:07:53:5E:81:83:D7:E8:DE:D7:89:D6:D6:A0:4D:FB:B9:7
6:8C:69:1A:13:4C:CA
 a=setup:actpass
 a=mid:video
 a=extmap:2 urn:ietf:params:rtp-hdrext:toffset
 a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
 a=extmap:4 urn:3gpp:video-orientation
 a=sendrecv
 a=rtcp-mux
 a=rtpmap:100 VP8/90000
 a=rtcp-fb:100 ccm fir
 a=rtcp-fb:100 nack
 a=rtcp-fb:100 nack pli
 a=rtcp-fb:100 goog-remb
 a=rtpmap:116 red/90000
 a=rtpmap:117 ulpfec/90000
 a=rtpmap:96 rtx/90000
 a=fmtp:96 apt=100
 a=ssrc-group:FID 194275495 2065741749
 a=ssrc:194275495 cname:9vOFkQsNuajNvEpF
 a=ssrc:194275495 msid:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7 f32658ea-37dc-
46e1-81ef-7b8ae53bd536
 a=ssrc:194275495 mslabel:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
 a=ssrc:194275495 label:f32658ea-37dc-46e1-81ef-7b8ae53bd536
 a=ssrc:2065741749 cname:9vOFkQsNuajNvEpF
 a=ssrc:2065741749 msid:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7 f32658ea-37dc-
46e1-81ef-7b8ae53bd536
 a=ssrc:2065741749 mslabel:eRSQZYs4B5k5w1DSgeb6LtD9ikfapvCotJZ7
 a=ssrc:2065741749 label:f32658ea-37dc-46e1-81ef-7b8ae53bd536
```

Primera Respuesta generada por parte del segundo navegador, del usuario bob. Esta primera respuesta será lo que es usuario bob establezca como descripción local. Se marca en rojo lo más destacado de dicha respuesta.

createAnswerOnSuccess
type: answer, sdp: v=0
o=- 5413457662029002875 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:CJW4F3C9ipS7H8bm
a=ice-pwd:+zZV7IAaycebJRDDrg/ngOaX
a=fingerprint:sha-256
F0:8D:79:77:1D:4B:04:E7:C0:5A:2C:D3:92:FF:86:03:C6:AE:7F:5A:8C:94:7E:DA:59:8
8:53:51:18:D0:66:53
a=setup:active
a=mid:audio
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=sendrecv
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10; useinbandfec=1
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:9 G722/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:126 telephone-event/8000
a=maxptime:60
a=ssrc:3227282109 cname:nLxldIqLF7ZupfWq
a=ssrc:3227282109 msid:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd 971137af-
520e-45a4-b609-7497d3c220db
a=ssrc:3227282109 mslabel:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
a=ssrc:3227282109 label:971137af-520e-45a4-b609-7497d3c220db
m=video 9 UDP/TLS/RTP/SAVPF 100 116 117 96
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:CJW4F3C9ipS7H8bm
a=ice-pwd:+zZV7IAaycebJRDDrg/ngOaX
a=fingerprint:sha-256
F0:8D:79:77:1D:4B:04:E7:C0:5A:2C:D3:92:FF:86:03:C6:AE:7F:5A:8C:94:7E:DA:59:8
8:53:51:18:D0:66:53

```
a=setup:active
a=mid:video
a=extmap:2 urn:ietf:params:rtp-hdrext:toffset
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:4 urn:3gpp:video-orientation
a=sendrecv
a=rtcp-mux
a=rtpmap:100 VP8/90000
a=rtcp-fb:100 ccm fir
a=rtcp-fb:100 nack
a=rtcp-fb:100 nack pli
a=rtcp-fb:100 goog-remb
a=rtpmap:116 red/90000
a=rtpmap:117 ulpfec/90000
a=rtpmap:96 rtx/90000
a=fmtp:96 apt=100
a=ssrc-group:FID 1764044322 635736251
a=ssrc:1764044322 cname:nLxldIqLF7ZupfWq
a=ssrc:1764044322 msid:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd 5ad76107-
661c-495e-b6e3-1cbeb516aee2
a=ssrc:1764044322 mslabel:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
a=ssrc:1764044322 label:5ad76107-661c-495e-b6e3-1cbeb516aee2
a=ssrc:635736251 cname:nLxldIqLF7ZupfWq
a=ssrc:635736251 msid:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd 5ad76107-
661c-495e-b6e3-1cbeb516aee2
a=ssrc:635736251 mslabel:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
a=ssrc:635736251 label:5ad76107-661c-495e-b6e3-1cbeb516aee2
```

Segunda Respuesta generada con los candidatos ICE. Dicha oferta será la que se mande en el mensaje OK. Esta oferta SDP será la que el primer usuario, p2pIPTV, establezca como descripción remota. Se marca en rojo lo más significativo de dicha respuesta.

```
createAnswerOnSuccess
type: answer, sdp: v=0
o=- 5413457662029002875 3 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
m=audio 50801 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
c=IN IP4 163.117.140.22
a=rtcp:9 IN IP4 0.0.0.0
a=candidate:183249902 1 udp 2122194687 163.117.140.22 50801 typ host generation 0
a=candidate:4136782018 1 udp 2122262783 2001:720:410:1020:74af:532:282c:27b0
34390 typ host generation 0
a=candidate:1474200709 1 udp 2122063615 163.117.223.127 49861 typ host generation
0
a=candidate:725943426 1 udp 2122131711 2001:720:410:3343:e0e8:d9bc:f23:442b
54283 typ host generation 0
```

```
a=candidate:1148180254 1 tcp 1518214911 163.117.140.22 0 typ host tcptype active
generation 0
a=candidate:3088167986 1 tcp 1518283007 2001:720:410:1020:74af:532:282c:27b0 0
typ host tcptype active generation 0
a=candidate:425556085 1 tcp 1518083839 163.117.223.127 0 typ host tcptype active
generation 0
a=candidate:1707538546 1 tcp 1518151935 2001:720:410:3343:e0e8:d9bc:f23:442b 0
typ host tcptype active generation 0
a=ice-ufrag:CJW4F3C9ipS7H8bm
a=ice-pwd:+zZV7IAaycebJRDDrg/ngOaX
a=fingerprint:sha-256
F0:8D:79:77:1D:4B:04:E7:C0:5A:2C:D3:92:FF:86:03:C6:AE:7F:5A:8C:94:7E:DA:59:8
8:53:51:18:D0:66:53
a=setup:active
a=mid:audio
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=sendrecv
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10; useinbandfec=1
a=rtpmap:103 ISAC/16000
a=rtpmap:104 ISAC/32000
a=rtpmap:9 G722/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:106 CN/32000
a=rtpmap:105 CN/16000
a=rtpmap:13 CN/8000
a=rtpmap:126 telephone-event/8000
a=maxptime:60
a=ssrc:3227282109 cname:nLxldIqLF7ZupfWq
a=ssrc:3227282109 msid:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd 971137af-
520e-45a4-b609-7497d3c220db
a=ssrc:3227282109 mslabel:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
a=ssrc:3227282109 label:971137af-520e-45a4-b609-7497d3c220db
m=video 50801 UDP/TLS/RTP/SAVPF 100 116 117 96
c=IN IP4 163.117.140.22
a=rtcp:9 IN IP4 0.0.0.0
a=candidate:183249902 1 udp 2122194687 163.117.140.22 50801 typ host generation 0
a=candidate:4136782018 1 udp 2122262783 2001:720:410:1020:74af:532:282c:27b0
34390 typ host generation 0
a=candidate:1474200709 1 udp 2122063615 163.117.223.127 49861 typ host generation
0
a=candidate:725943426 1 udp 2122131711 2001:720:410:3343:e0e8:d9bc:f23:442b
54283 typ host generation 0
a=candidate:1148180254 1 tcp 1518214911 163.117.140.22 0 typ host tcptype active
generation 0
a=candidate:3088167986 1 tcp 1518283007 2001:720:410:1020:74af:532:282c:27b0 0
typ host tcptype active generation 0
```

```
a=candidate:425556085 1 tcp 1518083839 163.117.223.127 0 typ host tcptype active
generation 0
a=candidate:1707538546 1 tcp 1518151935 2001:720:410:3343:e0e8:d9bc:f23:442b 0
typ host tcptype active generation 0
a=ice-ufrag:CJW4F3C9ipS7H8bm
a=ice-pwd:+zZV7IAaycebJRDDrg/ngOaX
a=fingerprint:sha-256
F0:8D:79:77:1D:4B:04:E7:C0:5A:2C:D3:92:FF:86:03:C6:AE:7F:5A:8C:94:7E:DA:59:8
8:53:51:18:D0:66:53
a=setup:active
a=mid:video
a=extmap:2 urn:ietf:params:rtp-hdrext:toffset
a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:4 urn:3gpp:video-orientation
a=sendrecv
a=rtcp-mux
a=rtpmap:100 VP8/90000
a=rtcp-fb:100 ccm fir
a=rtcp-fb:100 nack
a=rtcp-fb:100 nack pli
a=rtcp-fb:100 goog-remb
a=rtpmap:116 red/90000
a=rtpmap:117 ulpfec/90000
a=rtpmap:96 rtx/90000
a=fmtp:96 apt=100
a=ssrc-group:FID 1764044322 635736251
a=ssrc:1764044322 cname:nLxldIqLF7ZupfWq
a=ssrc:1764044322 msid:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd 5ad76107-
661c-495e-b6e3-1cbeb516aee2
a=ssrc:1764044322 mslabel:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
a=ssrc:1764044322 label:5ad76107-661c-495e-b6e3-1cbeb516aee2
a=ssrc:635736251 cname:nLxldIqLF7ZupfWq
a=ssrc:635736251 msid:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd 5ad76107-
661c-495e-b6e3-1cbeb516aee2
a=ssrc:635736251 mslabel:fp9Ev41Vq7Xh6VUwQP0Krc8r9uwQFs9uV6pd
a=ssrc:635736251 label:5ad76107-661c-495e-b6e3-1cbeb516aee2
```

Anexo D: Glosario de términos

3GPP: 3rd Generation Partnership Project
3GPP2: 3rd Generation Partnership Project 2
AAA: Authentication, Authorization and Accounting
AJAX: Asynchronous JavaScript And XML
API: Application Programming Interface
ASF: Apache Software Fundation
B2BUA: Back to Back User Agent
BGCF: Breakout Gateway Control Function
CDMA2000: Code Division Multiple Access 2000
CODEC: Codificador/DECodificador
CSCF: Call State Control Function
CSS: cascading style sheets
DOM: Document Object Model
DSL: Digital Subscriber Line
DTLS: Datagram Transport Layer Security
GPRS: General Packet Radio Service
GSM: Global System for Mobile Communications
HLR: Home Local Register
HSS: Home Subscriber Server
HTML: HyperText Markup Language
HTTP: Hypertext Transfer Protocol
ICE: Interactive Connectivity Establishment
I-CSCF: Interrogating Call State Control Function
IETF : Internet Engineering Task Force
IFC: Initial Filter Criteria
iLBC: Internet Low Bitrate Codec
IM-SSF: IP Multimedia Service Switching Function
IMS IP Multimedia Subsystem
IP: Internet Protocol
iSAC: Internet Speech Audio Codec
JSEP: Javascript Session Establishment Protocol
JSON: JavaScript Object Notation
JSP: Java Server Pages
JVM: Java Virtual Machine
MGW/MGCF: Media Gateway/ Media Gateway Control Function
MMUSIC: Multiparty Multimedia Session Control
MRFC: Media Resource Function Controller
MRFP: Media Resource Function Processor
NAT: Network Address Translation
OSA SCS: Open Service Architectura Service Capability Server
P-CSCF: Proxy Call State Control Function
PPT: Push To Talk
RAN: Radio Access Network
RCS: Rich Communications Services
RFC: Request For comments
RSVP: Resource Reservation Protocol

RTCP: RTP Control Protocol
RTP: Real-time Transport Protocol
S-CSCF: Service Call State Control Function
SDP: Session Description Protocol
SGML: Standard Generalized Markup Language
SGW: Signalling Gateway
SIP: Session Initiation Protocol
SIP-AS: SIP Aplication Server
SLF: Suscription Location Function
SMTP: Simple Mail Transfer Protocol
SRTP: Secure Real-time Transport Protocol
STUN: Session Traversal Utilities for NAT
TCP: Transmission Control Protocol
TISPAN: Telecommunications and Internet converged Services and Protocols for Advanced Networking
UA: User Agent
UDP: User Datagram Protocol
UMTS: Universal Mobile Telecommunications System
URI-SIP: Uniform Resource Identifier-SIP
VoIP : Voice Over IP
VoLTE: Voice Over LTE
W3C: World Wide Web Consortium
WAG: Wireless LAN Gateway
WEB-RTC: Web Real Time Comunication
WIMAX: Worldwide Interoperability for Microwave Access
WLAN: Wireless Local Area Network
XML: Xtensible Markup Language
XSLT: Xtensible Markup Language Transformations

Capítulo 8

Bibliografía

Bibliografía

[1] WebRTC. [En línea] http://www.webrtc.org/. Consultado por última vez Enero 2016.

[2]The Internet Engineering Task Force. [En línea] http://www.ietf.org Consultado por última vez Enero 2016.

[3]WebRTC [En línea] http://www.html5rocks.com/en/tutorials/webrtc/basics/ Consultado por última vez Enero 2016.

[4] WebRTC [En línea] https://www.webrtc-experiment.com/docs/how-to-use-rtcpeerconnection-js-v1.1.html#createOffer Consultado por última vez Enero 2016.

[5] Alan B. Johnston, Daniel C. Burnett. WebRTC: APIs and RTCWeb protocols of the HTML5 Real-Time Web.

[6] WebRTC [En línea] https://sites.google.com/site/investigacionwebrtc/codecs-de-audio Consultado por última vez Enero 2016.

[7] WebRTC [En línea] https://sites.google.com/site/investigacionwebrtc/codecs_video Consultado por última vez Enero 2016.

[8] Sons, John Wiley &. The 3G IP Multimedia Subsystem (IMS). 2008.

[9] The 3rd Generation Partnership Project. [En línea] http://www.3gpp.org/ Consultado por última vez Enero 2016.

[10] Aránzazu García García. PFC Desarrollo de un cliente de vídeo bajo demanda para redes con plano de control SIP/IMS. Octubre 2013.

[11] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, e. Schooler. SIP: Session Initiation Protocol RFC 3261. June 2002.

[12] V. Jacobson, C. Perkins, M. Handley,. SDP: Session Description Protocol RFC 4566. July 2006.

[13] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson,. RTP: A Transport Protocol for Real-Time Applications RFC 3550. July 2003.

[14] Java [En línea] https://www.java.com Consultado por última vez Enero 2016.

[15] JavaScript [En línea] http://librosweb.es/libro/javascript/ Consultado por última vez Enero 2016.

[16] JavaScript. Tutorial Javascript. [En línea] http://www.w3schools.com/js/ Consultado por última vez Enero 2016.

[17] HTML [En línea] http://definicion.de/html/ Consultado por última vez Enero 2016.

[18] HTML5 Introduction. [En línea] http://www.w3schools.com/html/html5_intro.asp Consultado por última vez Enero 2016.

[19] XML Tutorial. [En línea] http://www.w3schools.com/xml/ Consultado por última vez Enero 2016.

[20] Servlet [En línea] http://www.monografias.com/trabajos105/servlets-y-jsps/servlets-y-jsps.shtml Colsultado por última vez Enero 2016.

[21] . Apache Tomcat. [En línea] http://tomcat.apache.org/index.html. Consultado por última vez Enero 2016.

[22] JAIN SIP Specification. [En línea] http://jcp.org/en/jsr/detail?id=032/ Consultado por última vez Enero 2016.

[23]AJAX [En línea] http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/ Consultado por última vez Enero 2016.

[24] WebRTC [En línea] http://www.3cx.es/webrtc/que-navegadores-soportan-webrtc/ Consultado por última vez Enero 2016.

[25] COIT: El ingeniero de telecomunicaciones: Perfil Socio-Profesional [En línea] https://www.coit.es/index.php?op=publicaciones_detalle&idpublicacion=31 Consultado por última vez Enero 2016.

[26] J. Rosenberg Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC5245. Abril 2010

[27] J. Rosenberg,J. Weinberger, C. Huitema, Microsoft, R. Mahy, Cisco. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC3489. Marzo 2003

[28] W3C [en línea] http://www.w3c.es/ Consultado por última vez Enero 2016

[29] Acceso a Internet [En línea] http://www.rtve.es/noticias/20151001/79-hogares-espanoles-tiene-acceso-internet-mayoria-banda-ancha/1230482.shtml Consultado por última vez Enero 2016

[30] Mobile en España y en el Mundo 2015 [En línea] http://www.ditrendia.es/wp-content/uploads/2015/07/Ditrendia-Informe-Mobile-en-Espa%C3%B1a-y-en-el-Mundo-2015.pdf Consultado por última vez Enero 2016

[31] CISCO Visual Networking Index [En línea] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html Consultado por última vez Enero 2016

[32] Open IMS Core. [En línea] http://www.openimscore.org. Consultado por última vez Enero 2016

[33] Wireshark [En línea] https://www.wireshark.org/ Consultado por última vez Enero 2016.

[34] Gómez, Francisco Javier de Pablos. Desarrollo de un servicio P2P TV para redes de Internet de próxima generación. s.l. : ,Proyecto Fin de Carrera Universidad Carlos III de Madrid, Marzo 2012.

[35] H. Alvestrand Overview: Real Time Protocols for Browser-based Applications draft-ietf-rtcweb-overview-15 Enero 2016

[36] ETSI TS 122 228. Universal Mobile Telecommunications System (UMTS); IP multimedia subsystem;

[37] ETSI TS 24 229 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)

[38] Recommendation H.323 [En línea] http://www.itu.int/rec/T-REC-H.323-200912-I Consultado por última vez Enero 2016